Development of Machine Learning Applications: Named Entity Recognizer

Ghassan Abarbou

University of Tampere
Faculty of Natural Sciences
MDP in Software Development
M.Sc. thesis
Supervisor: Timo Poranen
May 2018

University of Tampere
Faculty of Natural Sciences
Degree Programme in Software Development
Ghassan Abarbou: Development of Machine Learning Applications: Named Entity Recognizer
M.Sc. thesis, 65 pages, 2 index pages
May 2018

Machine Learning is described in today's Information Technology world as one of the most promising research fields with great potential for providing a huge paradigm shift in modern systems. With the growth and the abundant availability of data, the need to structure, analyze and exploit these data has become a necessity for modern systems and a must for the major players within the field. Systems need to discover and structure data with minimal human involvement, while being able to adapt to the nature of the data, handle unseen patterns and still structure the data properly. One of the best-known applications of Machine Learning and one which output is considered the building block upon which more advanced systems rely is Named Entity Recognition. Named Entity Recognition (NER) is a classification task known better as one of the major applications of Natural Language Processing, which consists of classifying and assigning descriptive labels to sequences of text based on predefined classification categories.

The presented work aims at the conceptualization, design, implementation and evaluation of a system able to perform Named Entity Recognition on different datasets, with the maximum attainable performance by using the best result-yielding techniques and following the conventions of the field. The developed system implements a well-known statistical prediction framework proven to be best suited for classification tasks similar to NER; Conditional Random Fields (CRF) models were used to perform the initial recognition. Combined with the CRF models, the system developed different postprocessing methods to implement a Hybrid NER system oriented towards achieving performance levels comparable to the state-of-the-art literature in the field.

The research achieved language independent NER using the core of the developed system, and satisfying performance levels that were evaluated by conducting different experiments with different datasets and on different types of data.

Keywords: Named Entity Recognition, Conditional Random Fields, Information Extraction, Natural Language Processing, Hybrid NER, Datasets, Recognition, Features.

Contents

List of Figures

List of Tables

Acknowledgements

This work would not have been possible without the trust, support and guidance of my manager Aristotelis Kostopoulos, PhD. His ideas, comments and guidance throughout the project helped greatly in this journey; and for that I am immensely grateful. I am also grateful to my work teammates for their feedback and for being the best team I have ever worked with.

I would also like to thank Timo Poranen, PhD. for his guidance, valuable comments, understanding and patience during this project.

This work is dedicated to my family and friends; their love, support and encouragements were, and still are the light that shows the way.

1. Introduction

The field of machine learning is regarded nowadays as one of the most promising fields within the information technology (IT) world and research within this field is growing day by day. The machine learning trend is becoming omnipresent in almost all new applications within the IT world. From recognition systems to computational learning; every computer, mobile phone let alone other electronic devices include at least one, if not more applications that are based on machine learning. In simple terms, machine learning means teaching computers by providing known, expected output and making the computer learn its patterns. Then, based on what has been learnt, new processes are developed to deal with new input of the same kind [Rouse, 2016]. It is a branch of artificial intelligence that allows computers to learn without being explicitly programmed to do so; building programs and applications that can teach themselves how to interact with input based on the expected learnt teaching material [Rouse, 2016].

Within this paradigm, one of the most extensively studied branches is natural language processing (NLP). NLP is based on a combination of text mining (data mining in general) and the use of the machine learning paradigm to make robust systems that have decent performance [Nadeau, 2007]. The main task NLP is based on is assigning labels to words in a sequence of text, classifying them into defined target categories [Zuhori et *al.*, 2017]. This task has many applications in the field and amongst the most studied ones is Named Entity Recognition (NER).

Based on the need of deep low-level semantic analysis of text, NER is the foundation for many advanced information extraction systems [Poibeau, 2006]. The task consists of assigning labels to words in a text based on the function that the word holds within each sentence of the said text [Zuhori et *al.*, 2017].

Being considered one of the first steps of information extraction tasks, named entity recognition plays a major role in the mining of text to extract relevant information that will be later used as a basis to relaying solid grounds for data representation, linking and classification; leading to proper analysis of data semantics and consequently providing building blocks with which more advanced systems can build upon and harvest [Prasad et *al.*, 2015]. However, NER is not the absolute lowest level in

information extraction systems; it represents a high enough level that helps in understanding what is involved and how it is achieved within these systems.

The research field is considered as one of the most extensively studied subtasks of information extraction. There is a wide plethora of systems that implement NER using a variety of techniques achieving various levels of performance. However, a common concern in such implementations is the involved complexity and the near-dominant negligence of user friendliness when it comes to users who are not particularly research oriented and do not necessarily have previous experience or understanding of running such systems. This research aims to fulfill the need for an integrated proprietary system that is easy to set up and use. The setup of the research comes from the future orientation and vision of the company I am currently working for. Following today's market trends, the company is moving towards providing machine intelligence solutions. Based on this vision, the need of a proprietary system that will handle NER with decent performance and user friendliness became apparent and was particularly intriguing to me as a research topic.

The research examines the most widely used algorithms and techniques to build an integrated named entity recognition system for different languages, evaluate its performance and improve on it. Then, build an interface that will expose the main functionality provided by the engine in a user-friendly framework improving on the usability of such systems. This research will be part of a thesis working position with my current employer as an addition to the company's portfolio of tools oriented towards machine learning and machine intelligence.

The presented work starts by examining the machine learning field and the software engineering processes to get familiarized with the basics of the field. It will then proceed to focus on one major area of activity, which will be named entity recognition as it is considered to be the basis of many information extraction systems and its output is used within more complex systems.

NER analysis shall begin by identifying the exact paradigm that will be used to achieve it. First, the feasibility of the system will be studied and closely examined to identify the practical scope that the system will operate on. Then, the study will move on to applying proper software engineering processes to identify the most adequate

software life cycle suitable for the project. After the analysis and design of the system, the study will move on to finding, analyzing and processing proper corpora for NER and the implementing of the statistical prediction model module of the system. Within this step, the most efficient algorithms will be implemented using a suitable machine learning framework.

The research will then shift to balancing the datasets (training, validation and testing) and training the model using the training data; then processing the testing and the validation (if needed) sets. The final step within the NER system will be to assess the performance of the trained model, analyze it and work on improving it until satisfactory performance metrics are reached. The research tackles the conception, design and implementation of interfaces that will make use of the developed NER system with additional features adding value to it. After development, the interfaces will be tested and evaluated, and the added value they bring will be reflecting on. The main target language of the development stage for the research will be English due to the abundance of the relevant data and the availability of a solid research base. However, throughout the research, the language independency aspect will still be one of the main focuses of the project as this is one of initially set goals. Scalability of the developed modules to handle different languages and achieve decent performance metrics for other languages will be checked and evaluated as the research progresses.

Ideally, the application would go through a normal software engineering product life cycle to have an end product that can be evaluated. However, to accommodate for the time and effort spent on researching unfamiliar machine learning practices that are involved within NER systems; as well as the time needed to deal with the huge amounts of data that will be used within the developed system, adaptations had to be made. The project followed a modified scrum methodology that accommodated for the above-mentioned project related characteristics.

The need for this research arose from the business orientation of the work place and the general direction it is taking. The company is working on multiple machine intelligence fronts and needs proprietary systems to cover different applications related to this orientation. The idea behind the research was conceived within this need and this context. Hence the need for an integrated system that will perform language

independent named entity recognition on a large scale implementing state-of-the-art techniques and approaches, and reaching the best attainable results in terms of performance and scalability. A proprietary system that will cover an aspect of machine learning that is considered the basis for most of information extraction systems; a system that is easy to use, easy to set up and scalable to different languages and different types of tasks.

The study aims at conceptualizing, designing, implementing and evaluating an integrated named entity recognition system with language-independent reusable subparts. The core of the system will be a machine learning engine that will be able to perform language independent NER; coupled with this core module there will be language-specific rules and components that will change from language to language. Together with the engine, they shall constitute an operational named entity recognition system satisfying the Hybrid NER paradigm.

The main objectives of the research will be to match metrics of the majority of the current systems resulting from the latest research on the field. Investigating what is used, how it is used and the best ways to combine it to achieve the best results will be the core of the research. Consequently, the research questions that this study will be answering are as follows:

- What are the most widely used algorithms and approaches within the field of Named Entity Recognition and how can they be optimized and used in this specific context?
- What are the software engineering processes used to improve efficiency in Named Entity Recognition and how to use and combine them for better metrics and performance?

The next section covers the related work and sets up the theoretical framework of this thesis. Section 3 illustrates the developed NER system architecture and describes in detail the different modules that the system is composed of. Section 4 goes over the experiments and the phases of the project with details of the used methodology. The results of the research are synthesized in Section 5; and the thesis concludes with subsections summarizing the thesis, going over the limitations and introducing the future work proposed to mitigate these limitations.

2. Literature Review

2.1. Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying and classifying words or phrases in a text (referred to hereafter as entities or named entities) according to rigid designators defined by the actual target purpose of the task [Nadeau, 2007]. The conventional designators include Person, Location, Organization and most commonly a miscellaneous type to accommodate for various other types that do not necessarily fall within these three conventional categories [Brychcin et *al.*, 2015]. NER is a prominent research field within machine learning due to the fact that it is considered to be the starting point for many of the bigger and more complex machine learning and information extraction based applications [Tjong and De Meulder, 2003]. NER aims at extracting and classifying labels in text, such as proper names, biological species, quantitative words or more inclusively, language and domain specific expressions [Tjong and De Meulder, 2003]. This is particularly important in identifying the entities within the text based on the context that they occur in; making the system more robust when faced with unknown similar input. This allows the NER systems to identify the input more accurately and produce a good semantic analysis base that other information extraction applications can rely on [Grishman and Sundheim, 1996]. Such applications include: improving search engines and search engine queries; monitoring trends in textual data that are made available every day by individuals, organizations and governments all over the world; and building user adapted and oriented applications based on users' behavior and historic data logs. In addition, it is widely used in biology and genetics [Nadeau, 2007]. The following is an example of a text marked with four types of entities (Person, Location, Organization, and Date):

In <**Date**> 1895 <**/Date**>, at the age of 16, <**Person**> Albert Einstein <**/Person**> took the entrance examinations for the <**Organization**> Swiss Federal Polytechnic <**/Organization**> in <**Location**> Zürich <**/Location**>

Named Entities (NEs) are aimed to designate only entities that are rigid designators, which include proper names and certain natural terms but only when used in a specific context [Nadeau and Satoshi, 2007]. "Named" defines a restriction applied to the classification of words or phrases (entities) where only entities that can be described by

one or more rigid designators are considered and classified accordingly [Nadeau and Satoshi, 2007]. For example, in the sentence “The University of Tampere is a good university”, the word (token) “University” occurred twice. In the first occurrence it is considered to be part of the composite entity “University of Tampere” (Organization). However, the second occurrence of the word “University” is not considered an entity. Similarly, the word “Tampere” is also viewed as part of the entity “University of Tampere” (Organization); whereas, in a different context it will be marked as a Location entity. Similarly, based on the context or the goal of the task, there may also be NEs that are categorized as invalid. NEs are viewed as invalid when they do not fit the general aim of the task or the intent of the defined designators [Kripke, 1982].

The concept of named entities was defined as early as the 1990s. It started as a broad definition where NEs were defined as “unique identifiers of words” and included mostly company names. Company names were considered problematic in natural language processing due to the fact that they were mostly foreign words and abbreviations. In early 2000s, the term was narrowed down to “a proper noun, serving as name of something or someone” used to classify unknown objects into known categories that are aimed at solving a certain problem. By 2007, the proposed definition of NEs elaborated on this to characterize them as labels or a group of labels referring to one or more rigid designators. [Marrero et *al.,* 2013] Rigid designators are defined as terms designating “the same object in all possible worlds in which that object exists and never designates anything else” [LaPorte, 2016]. Though the definition of NEs differed from research to research and from era to era, the main aim and general idea remains the same. Named entities are labels or groups of labels designated to categorize and classify a token or a group of tokens within a sequence of text depending on the context in which they occur. They completely depend on the context within which they happen (their role within the sentence) and on the aim of the task at hand. If for example, the task is to extract and label names of proteins within a scientific text the conventional person, location, and organization designators will not be considered. In the context of NER, these rigid designators are referred to as labels, tags or classes. For simplicity, rigid designator thereafter will be referred to as labels.

On the other hand, there are labels that can categorize more than one type of entities depending on the context, NE structure or reference. Such NE types are called ambiguous types and are one of the main challenges when dealing with named entity recognition [Kuperus et *al.*, 2013]. Ambiguous types can be classified into three main categories.

1. Semantic: where it is hard to classify the NE based on its semantics [Kuperus et *al.*, 2013]. Let us consider the example of the word "Paris" in the two sentences: "I visited Paris last fall", and "Paris was an inventor". In the first, the word or token Paris is a location NE type referencing the city of Paris, so it is categorized as such; in the second the "Paris" is a proper name NE type and references the Paris (person) entity. The NE type in these two sentences can be concluded from the context. However, in a sentence such as: "I like Paris". The type cannot be inferred from the context; hence the complexity of the ambiguous NE type in the last instance.
2. Structural: where the NE boundaries are to be defined, how they differ depending on the context as well as the structure of the entity itself and how to decide what to include and what to leave out [Kuperus et *al.*, 2013]. An example would be, the expression "Ouiouane Lake" where it is not clear whether the "Lake" token is part of the entity or not. Within this research such entities will be referred to as composite entities.
3. Reference: where the category to which the NE belongs may differ from context to context and from task to task [Kuperus et *al.*, 2013]. For example, in a task that includes classifying addresses, the token Tampere is a location but within an address of one of the city's streets it is classified as part of an address NE.

There exist many extensive studies in the field of named entity recognition and the field is described as a mostly solved prominent subtask within natural language processing and information extraction. However, there is always room for efficiency and performance improvements as well as including support for different languages not so widely studied as English, German, Chinese, Spanish and French [Marrero et *al.*, 2013].

Named entity recognition conventionally utilizes two different approaches: the rule-based/dictionary approach and the machine learning approach. The rule-based/dictionary approach performs recognition using rules, dictionaries or other lists

that are hand coded, collected, and formulated by human annotators [Prasad et *al.*, 2015]. This requires huge amounts of human effort; hence the need for other alternatives. The second approach which is based on the machine learning paradigm is characterized as being highly automated and as considerably reducing the required human effort and involvement. This approach has two main forms: supervised and unsupervised learning. [Prasad et *al.*, 2015] The unsupervised learning does not use training data to train models and do the recognition but relies entirely on clustering, lexical patterns and statistics based on large unannotated data [Nadeau, 2007]. The supervised learning approach is based on training a model that learns from manually annotated data. The model is built as a statistical model, based on the relations between each word/token, its annotation (label) and its context. Then, based on that model, predictions are made on raw input by adding the labels to the input data [Prasad et *al.*, 2015]. Among the most used statistical model generation methods we find the Hidden Markov, Maximum Entropy, Support Vector Machine and Conditional Random Fields models [Tjong and De Meulder, 2003]. An additional technique used is the semi-supervised learning where a small amount of annotated data is used, combined with a larger amount of unannotated inputs. The annotated data are used to start the learning process where the recognized patterns are used to find similar patterns in the larger dataset and extrapolate on the findings. This technique is fairly new and yields inferior results to the supervised learning [Nadeau, 2007]. The third approach to named entity recognition is called the Hybrid NER approach and it is a combination of the rule-based/dictionary and the machine learning approaches.

The need for NER comes from the abundance of data in the form of digital information from the Internet. Such information mainly includes user-generated data from social media platforms or other similar mini-blogging interfaces. Mining this information is becoming a necessity in accordance with the current trends based on the need to discover information and manage it in information extraction systems. Developing methods to structure unstructured data is becoming an essential aspect of information management, and NER is crucially being the starting point where semantic analysis is applied to unstructured data, classifying it into predefined atomic categories.

Named entity recognition is particularly useful in a plethora of information extraction tasks. Some of these tasks include [Marrero et *al.*, 2013]

- Semantic annotation that aims to identify concepts within the input and relations between them.
- Question answering systems designed to clarify and answer queries.
- Semantic web and ontology analysis conducted for the task of classifying information into ontology classes that are further used to make information interoperable across the input.
- Social web and opinion mining where the aim is to study general trends and preferences based on the social media texts and opinions.

Figure 1 illustrates an example flowchart of the role that NER plays within text mining and information extraction systems. In the flowchart, NER comes at early stages of the information flow of such systems providing low-level semantic analysis of the input. It also makes use of the lower-level analysis processes such as tokenization and gazetteer output. The classification output from NER is for co-reference resolution identifying elements based on hierarchies from the defined grammar rules. This is built upon further to ultimately reach the final aim of the system to provide ontology classes for the input.

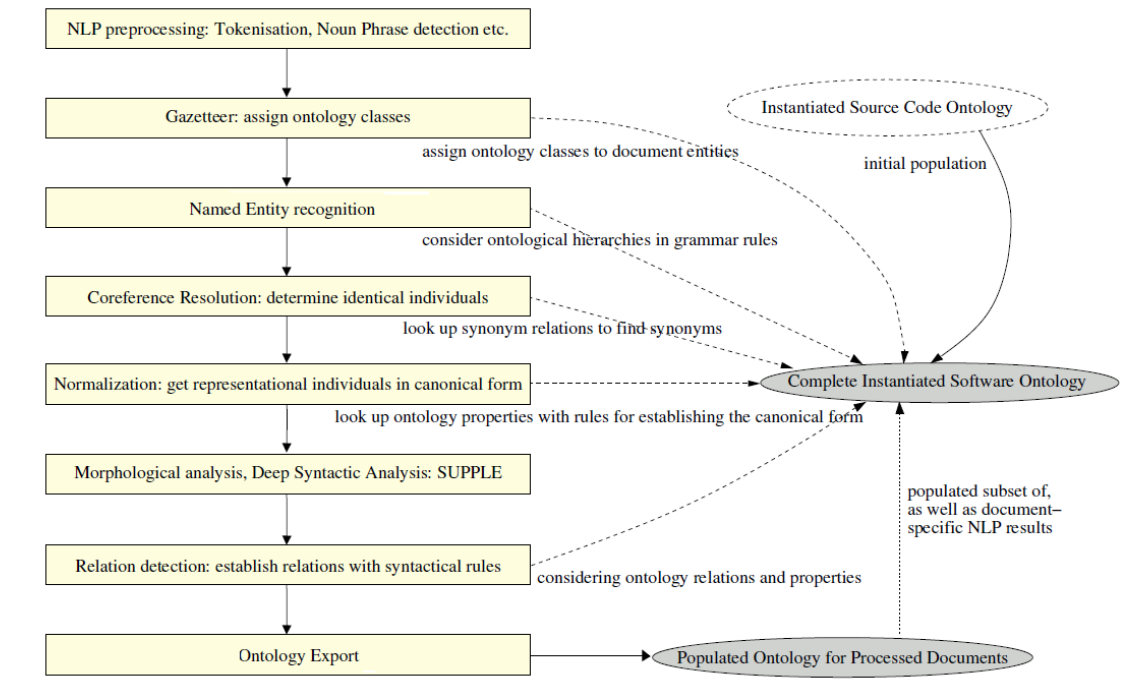

Figure 1. Example of the workflow of a text mining system [Kedad et *al.*, 2007].

2.2. Theoretical Framework

Subsections 2.2.1 to 2.2.6 cover the definition of the important components and concepts that set the theoretical framework of this research. The subsections explore the previous related work done within each subfield and focus on the concepts for which the initial findings proved to be the best result-yielding techniques that will lay the basis for this project's experiments. Subsections 2.2.7 and 2.2.8 cover the conventional evaluation methods used to evaluate NER systems and define the specific metrics used to evaluate the developed system.

2.2.1. Rule-based NER

Rule-based NER defines rules that are applied to the input classifying it into the relevant categories. The rules are handcrafted by a linguist and implemented to extract patterns that are used to identify and classify NEs [Poibeau, 2003]. This is achieved by starting from the assumption that the rule contains the name pattern of the entity in the input which is then used to identify the entity. The performance of systems relying on the rule-based approach is in direct correlation with the quality and inclusivity of the handcrafted rules, and it is very domain-specific [Kedad et *al.*, 2007]. Consequently, it requires a lot of manual effort and heavy human involvement which translates to time and cost. Techniques used within this approach vary, but the main goal is to make a decision on the classification of a word based on a linguistic or a domain restricted pattern that fits within the defined rules [Poibeau, 2003].

2.2.2. Dictionary-based NER

The dictionary based approach is based on list lookups. In this approach, lists of accepted named entities are compiled and categorized, then the input is compared with these lists and a matching method is developed; resulting in the assignment of labels to the input text based on the results of the matching [Prasad et *al.*, 2015]. Within this approach the same entity may be categorized under multiple types; consequently, a matching method is needed to decide which NE to keep. This approach, if carried out alone, is marked by a deficiency in performance due to the ambiguous types and the fact that the whitelists have to be either manually compiled and verified or scripted on huge dumps of data to extract adequate lists in terms of size and variety of types [Prasad

et *al.*, 2015]. Another complication that may arise with the use of this approach is the amount of data involved and how the list lookups will handle it. Conventionally, the used lists are referred to as lexica, gazetteers or whitelists and they all refer to accepted NE lists that are handcrafted and used for matching the input and providing labels for the matching words in the input.

2.2.3. Supervised Learning

Supervised learning is the most widely used and amongst the better performing approaches in named entity recognition.

As early as its definition in the sixth Message Understanding Conference (MUC-6) and with the first encouraging results in the reference CoNLL 2003 shared task: language independent named entity recognition by [Tjong and De Meulder, 2003], the NER task has always been viewed and addressed as a machine learning problem that has been proven to have better performance with supervised learning. Most of the systems participating in the CoNLL 2003 shared task used supervised learning as the main approach to achieve NER with precision levels ranging between 71% and 88.9%. Since in NER, as in other natural language processing tasks, the main goal is to achieve the best possible results, the vast majority of the systems or at least the best performing ones nowadays rely on the supervised learning approach [Neumann and Xu, 2004].

Multiple factors make named entity recognition impractical and less efficient when relying on other conventional approaches without incorporating a machine learning component into the recognition. Briefly, these reasons include the following:

- The numbers of target-fitting NEs are most often too large to include in lists [Neumann and Xu, 2004].
- Named entities, being proper nouns do not have a unique form, which also keeps changing [Neumann and Xu, 2004].
- Abbreviation and acronyms are hard to recognize without context pattern matching rules [Nadeau, 2007].
- Pattern-matching handcrafted rules are hard to formulate and very domain-specific [Poibeau, 2003].

- Named entity boundaries are very hard to precisely identify with traditional methods [Neumann and Xu, 2004].
- Traditional methods produce ambiguous types, which lowers the performance of systems relying entirely on them [Marrero et *al.*, 2013].

Consequently, the use of machine learning and specifically supervised learning to perform NER is a dominant solution in the field [Nadeau, 2007]. Supervised learning is defined as a sequential prediction problem [Gao et *al.*, 2017]. The prediction is made on the introduced input based on observational known (observed) data by building a statistical prediction model [Gagné, 2013]. For NER, the main goal behind using supervised learning is the classification of new input based on the learnt data [Kanya and Ravi, 2013]. Figure 2 is a simplification of the principle upon which supervised learning is based. The figure shows how in supervised learning, known data and known response are used to train a model which is then used to predict new responses for the input new data.

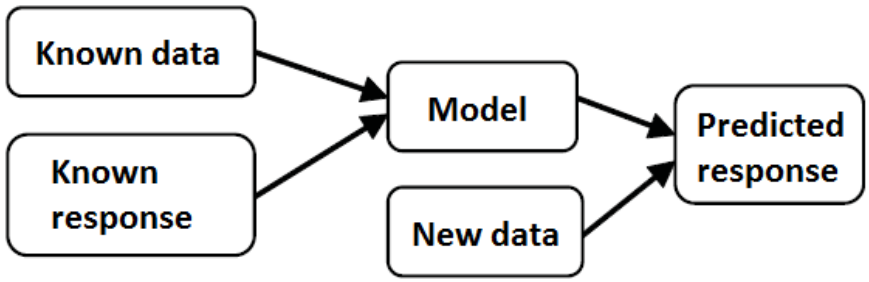

Figure 2. Supervised learning illustration [Kanya and Ravi, 2013].

In supervised learning, the aim is to "optimize a model from observations depending on a performance criterion" [Gagné, 2013], where observations are patterns and valid occurrences presented in the large amount of data that the model is trained on. Supervised learning is formally defined as

$$y = h(x|\theta)$$

where, **y** is the associated value as output, **x** is the observation as input, **θ** is the model parameters and **h()** is the general model function [Gagné, 2013].

Systems using this approach read a large amount of annotated data that illustrates the classification problem at hand, learn the patterns within the dataset and predict the

output based on the observations from the learnt data patterns. In the case of supervised NER, the input is a large corpus that typically represents the tokens (words) and their corresponding labels identifying the NEs. The model is then trained on the corpus to learn the labels and the context within which they occur, memorizes the entity lists, creates different disambiguation rules out of the extra information from the tokens (called features, to be covered in the next sections) and aggregates the information (observations) in a statistical model. Based on this model, predictions are made on similar input. [Gagné, 2013]

Primitive supervised learning systems recognize a named entity from the testing or validation sets only if it was learnt in the training set as an entity [Nadeau, 2007]. However, with the extensive research and improvement to the field and the usage of appropriate statistical prediction techniques, modern supervised leaning systems involve a plethora of variables that make such systems' performance decent. The mentioned techniques include prediction algorithms, probabilistic frameworks and feature-based learning [Chang et *al.* 2011]. This, grants NER systems implementing this approach the ability to "recognize previously unknown entities" [Nadeau, 2007] within the input which is the absolute core of NER.

Among the most studied and applied techniques within supervised learning we find:

- Hidden Markov Models. [Bikel et *al.*, 1997]
- Decision Trees. [Satoshi, 1998]
- Maximum Entropy Model. [Borthwick et *al.*, 2002]
- Support Vector Machines. [Masayuki and Matsumoto, 2003]
- Conditional Random Fields. [Lafferty et *al.*, 2001]

Based on multiple studies on supervised learning [Gao et *al.*, 2017; Gagné, 2013] and its application in named entity recognition [Neumann and Xu, 2004; Ratinov and Roth, 2009; Chang et *al.* 2011], one of the appreciated and most used techniques that serves the needs for the classification aspect of named entity recognition particularly well is Conditional Random Fields. Consequently, the research focuses on this specific technique as a statistical prediction basis for the machine learning module of the system.

2.2.4. Conditional Random Fields

Conditional Random Fields (CRF) is a probabilistic framework for labeling and segmenting sequences of data. The CRF model is built as an exponential model determining the conditional probability of sequences of labels given the complete observation sequence. A Conditional Random Field is an undirected graphical model where a "Conditional Field" is constructed for a pair of random variables representing respectively the observations and the labels sequences which is globally conditioned on the whole observation sequence. [Lafferty et *al.*, 2001; Wallach, 2004]

A CRF model is based on determining the distribution of a set of random variables constituting the vertices of a graph, where the edges are the dependencies between each pair of the set of the two random variables [Chang et *al.* 2011]. Formally, Conditional Random Fields are defined as follows by [Lafferty et *al.*, 2001]: assume two sets of random variables **X** and **Y** over sequences of observations and labels respectively. In the case of NER, every element of **Y** (Y_i) belongs to a finite set of labels and every element of **X** (X_i) belongs to the set of human language sentences. Letting a conditional model be **p(X|Y)**, and given an undirected graph G with vertices **V** and edges **E**, **G=(V, E)** where **V** index the element of **Y**, in a Conditional Random Field **(X,Y)** being conditioned on **X,** each random variable $\mathbf{Y_i}$ with respect to **G** satisfies the following

$$p(Y_i|X, Y_q, i \neq q) = p(Y_i|X, Y_q, i \sim q)$$

where **i** and **q** belong to **V** and are neighbors [Lafferty et *al.*, 2001]. The neighbors of a node from **G** are vertices from **V** that are adjacent to the said node [Gassert, 2017].

The graph **G** can take any arbitrary form given that it represents the dependences in **Y** but when modeling sequences, the simplest encountered form is a first-order chain form illustrated in Figure 3, where the set X of nodes corresponds to any of the elements of Y in a first-order chain.

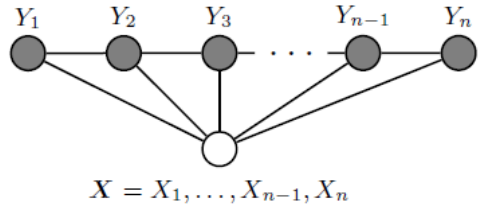

Figure 3. Graphical representation of a chain CRF.

The conditional probability of the Conditional Random Field **(X, Y)** is defined as the normalized product of the feature function and it is computed as follows [Wallach, 2004]:

$$p(Y|X,\lambda) = \frac{1}{Z(X)} exp(\sum_{i=0}^{n} \sum_{j} \lambda_j f_i(Y_{i-1}, Y_i, X, i)) \quad (2.4)$$

In the above, $f_i(Y_{i-1}, Y_i, X, i)$ is the feature function with either numerical or binary values. The feature function is expressed on a set of real-valued atomic or empirical characteristics **b**(X,i) of the elements of the observation X. Each element from the observation is marked using these values. For example, **b**(X,i) can be expressed on an element of X as follows

$$b(X,i) \begin{cases} 1 & \text{if at the position } i, X \text{ has the token "John"} \\ 0 & \text{otherwise} \end{cases}$$

Each feature function is then defined on the values of **b**(X,i) as follows

$$f_i(Y_{i-1}, Y_i, X, i) \begin{cases} b(X,i) \text{ if } Y_{i-1} = \text{S and } Y_i = \text{S\_PERSON} \\ 0 \quad \text{otherwise} \end{cases}$$

Moreover, **λj** is the feature-learning parameter over the observation X, representing the weights of the corresponding feature function [Nongmeikapam et *al.*, 2011]. **Z(X)** is a normalization factor defined as [Wallach, 2004]:

$$Z(X) = \sum_{y} exp(\sum_{i=1}^{n} \sum_{j} \lambda_j f_i(Y_{i-1}, Y_i, X, i))$$

This shows why CRF models are a widely used learning algorithm for NER; they do not only consider the probability of a word having a label as a standalone, but as part of the whole observation sequence (sentence), while considering the word before and after it and its context within the sentence. Figure 4 shows a simplified illustration of

how the probability is computed within a sequence using a CRF model where the probability of a token having a label is based on multiple connections between the adjacent tokens and labels

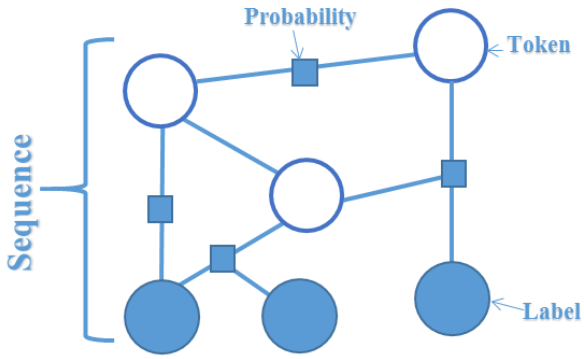

Figure 4. Illustration of CRF probability calculation.

For NER and other systems using CRF as a statistical prediction model, the goal is to maximize the conditional probability in (2.4). Solving a CRF is based on the resolution and estimation of the λ_j feature-learning parameter. The product of (2.4) over all of the training data (observation X) in reference to λ_j is referred to as the log-likelihood. The log-likelihood function is a concave function, which guarantees convergence to the global maximum [Wallach, 2004]. The most widely used methods to determine the feature-learning parameter are based on using a gradient descent algorithm, an iterative scaling or a Quasi-Newton method [Chang et *al.* 2011; Wallach, 2004].

As observed, the feature function is a major factor in determining the conditional probability of a word having a label within a sentence. NER features are crucial components of any system using CRF models; they are aimed at characterizing the word within the sentence and determining its form, nature and role.

2.2.5. NER Features

NER using CRF relies heavily on the features to distinguish words and infer their context. Features play a major role in creating the disambiguation rules when the model is generated and they can be seen as the most crucial aspect of CRF models. Features are defined as describers or characteristic attributes of words that help better define the

role of the word within the sentence and context. For example, features can include the case of a token (upper case, lower case or mixed), POS (part of speech) tags that define the grammatical function of the word within the sentence, the word's root, internal or external (final) punctuation and many more features targeted at improving the efficiency. [Nadeau, 2007]

For NER, features must be selected carefully as they play a major role in the recognition. They can be categorized into two main types: language-dependent and general features. Language-dependent features, as their name suggests, are language-specific and describe a specific aspect of the word within the input. For example, the stem of a token is considered language-dependent which makes features based on stemming language-dependent as well. General features determine the general form of the word based on its apparent aspect, such as the lexical form, the morphological form or the nature of the word or token [Luo et *al.*, 2012]. For example, whether a token is capitalized, is a number, or is a punctuation or not are considered general features.

Formally NER features can be split further into the following categories [Ram et *al.*, 2010; Benajiba et *al.*, 2008]:

- Context base, which mark the context of the token within the sentence. They help the CRF learn the word and the syntactic information of NEs.
- Word-based or morphological, which mark the nature of the word. This type of feature aids in identifying the nature of the word being for example nominative, dative, possessive, numerical, directional, locative and so on.
- Structural or sentence-based, which mark the position and the role a word plays in a sentence. For example, if a noun is preceded by a verb, the noun is a probable NE candidate and as marked as such for the CRF training.

2.2.6. Hybrid NER

When carried out individually, all approaches to NER show deficiencies. They either require considerable amount of human involvement, large amounts of data or trade-offs in performance for overcoming the information availability and access bottleneck [Silva et *al.*, 2006]. To mitigate these limitations and keep the desired automation aspect of NER especially using the machine learning approach, most of current research findings

suggest the use of combinations of approaches to improve the performance of machine learning based systems. Since such systems have the ability to recognize previously unseen entities while retaining decent performance, combinations of classifiers, handcrafted rules and the use of lexica are widely used in machine learning based systems in what is referred to as the Hybrid NER approach [Chiong and Wei, 2006].

For languages with especially complicated morphologies and sentence structure and for noisy data (unedited data with un-reviewed user-generated text), using classifiers based only on statistical prediction imposes certain restrictions and consequently lowers performance [Benajiba et *al.*, 2008]. To overcome these limitations and maintain the main goal of such systems, i.e. having the best performance metrics possible, a plethora of techniques are used. Among these we find the combination of multiple text classifiers generated by different prediction algorithms to compensate for the limitations of each other and refine the results [Silva et *al.*, 2006], as well as the combination of the classification results from the machine learning model with handcrafted rules to identify grammatical patterns [Chiong and Wei, 2006]. However, the technique that yields the best results according to the research in the field consists of combining the three techniques and approaches covered in Subsections 2.2.1, 2.2.2 and 2.2.3.

Namely, the best practice in this context is to combine the results from list lookups, with the results from the statistical prediction model along with selective labeling using the handcrafted rules. This is achieved by adding a postprocessing step where the results are combined and the labels are determined based on weights, confidence values and ambiguity-resolving results [Meselhi et *al.*, 2014]. The developed NER system opts for the later technique of combining rule-based/dictionary and the machine leaning based approaches for performing and refining the recognition, making the system a Hybrid NER system.

2.2.7. System Evaluation

Within the machine learning paradigm, conventional metrics are always taken into consideration when evaluating systems. This research followed this convention and the agreed upon metrics were used to evaluate the developed system. NER systems traditionally adopt relatively unified evaluation methods that aim at determining how

performant the evaluated system is in classifying the input and recognizing the NEs and their corresponding labels. To evaluate a NER system, generally the testing or validation set is processed. Two versions are kept of the same set, one with original labels from the corpus (gold standard) and one that was stripped of those labels and underwent the recognition process adding the predicted labels to the stripped set (Figure 9) [Atdağ and Labatut, 2013]. The two lists are then compared, resulting in traditional machine learning counts that then take part in calculating the main metrics used to evaluate the system. The classification counts that are involved in the calculations of the system evaluation metrics aim at comparing the recognized NEs against the gold standard NEs [Finkel et *al.*, 2005]. The counts categorize NEs that are actual NEs, falsely recognized tokens and unlabeled NEs. The counts are formulated by counting the following [Atdağ and Labatut, 2013]:

- True Positive (TP): an actual NE that was recognized as such for the respective token or group of tokens.
- True Negative (TN): an unclassified token that is not an actual NE.
- False Positive (FP): an NE recognized by the system that is not an actual NE.
- False Negative (FN): an actual NE that was not recognized by the system.

These counts are summarized into a prediction summary in a tabular form called a confusion matrix [Salama et *al.*, 2015]. A confusion matrix is used to determine the type of errors the classifier might be making and on which exact classes [Brownlee, 2016]. The confusion matrix is conventionally for two-class classification and is represented as follows [Salama et *al.*, 2015]:

		Predicted	
		Positive Class	Negative Class
Actual	Positive Class	TP	FN
	Negative Class	FP	TN

Table 1. Confusion Matrix.

Traditionally, counts are obtained by comparing the original golden standard to the predicted labels on the same position, in what is called spatial comparison [Atdağ and Labatut, 2013]. Since the system was evaluated against the reference literature, which mostly uses the spatial comparison along with the exact match method, where no partial credit is given to partial matches for composite entities, both exact match and spatial

methods were used for evaluation in this work. An example of this, the reference CoNLL NLP [Tjong and De Meulder, 2003] evaluation script which was used to evaluate all the systems participating in Coling 2016 [Ritter et *al.*, 2016] (Section 4.4) had spatial and exact match evaluation. During all phases of the project more than two classes were targeted for all the experiments. Phases I and II had the traditional person, location organization classes; the noisy data analysis had two variants, one with 10 classes and one with two classes. Given that the above introduced counts and the metrics based on them covered in the next Subsection are binary or two-class defined, a multi-class classification was used. One-versus-all (OVA) [Aly, 2005] was applied where for the experiments having more than two classes the evaluated class was the positive class from the confusion matrix (Table 1) and all other classes were considered as the negative class.

2.2.8. Accuracy, Precision, Recall and F-measure

Once the two datasets are compared, the confusion matrix counts are used to compute set-level generalized performance measures that determine how well the system is performing in terms of classifying the input and detecting the NEs. The two main distinct measures are precision and recall which are combined to represent the F-measure of a system. To these three, accuracy can be added as a secondary measure. However, accuracy (as observed in Section 5 with accuracy of 90% and above even for the problematic types) within NER does not convey a lot of meaning since it does not reflect what kind of errors the classifier is making and how well the classifier is categorizing the tokens into their correct classes [Brownlee, 2016]. Consequently, in NER the main metrics are precision and recall and F-measure. The four metrics can be defined as follows [Atdağ and Labatut, 2013]:

- **Accuracy:** Percentage of correct predictions (tokens that are not NEs are recognized as not NEs).
- **Precision:** Percentage of NEs that were recognized (positives) and were correct.
- **Recall:** Percentage of actual NEs that were recognized and were correct.
- **F-Measure:** Mean of precision and recall.

Formally and using the previously defined counts the measures are computed as follows [Atdağ and Labatut, 2013]:

- $Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$
- $Precision = \frac{TP}{TP + FP}$
- $Recall = \frac{TP}{TP + FN}$
- $F - Measure = \frac{2(Precision * Recall)}{Precision + Recall}$

2.2.9. User-Generated Noisy data

User-generated data as the name suggests, are data generated by users over the internet [Marinho de Oliveira et *al.*, 2013]. The main source of such data is micro-blogging activities that find infrastructure and are made available to the masses through platforms such as Facebook and Twitter [Ritter et *al.*, 2011]. The abundance of these data, as millions of user-generated entries are circulated daily within the mentioned platforms, raised the need to exploit it. With the growth of such data in size, their global aspect and their relevance; the need to analyze, structure and classify them became a necessity in the modern knowledge discovery systems. As covered before, NER is the basic component of such system. However, due to the nature of the language used in these platforms and the source of the data, challenges arise in NER tasks on such data [Ritter et *al.*, 2011]. Challenges that include [Marinho de Oliveira et *al.*, 2013]:

- The large amount of data that can be hard to stream, store and process.
- The lack of contextualization and formality: the entries (statutes, tweets) in most cases are personal thoughts and inside exchanges that only the user knows the context of; and that more often than not, lack proper sentence structure, capitalization and punctuation.
- Language diversity and errors: within the same entry there might be words belonging to multiple languages, and likely misspelled words.

3. NER System Architecture and Modules

3.1. Architecture

Most information extraction systems are based on the premise that input files are introduced, formatted into an acceptable format and processed; then output files are produced. This research did not stray from this conventional structure. The developed NER system takes as input text files of sentences then formats them to the needed format depending on the processing that those files are to undergo. The resulting formatted file is then handled using the corresponding system modules. The end results are processed files with formatting similar to the input for uniformity.

NER systems' architecture can be conceptualized using the Figure 5.

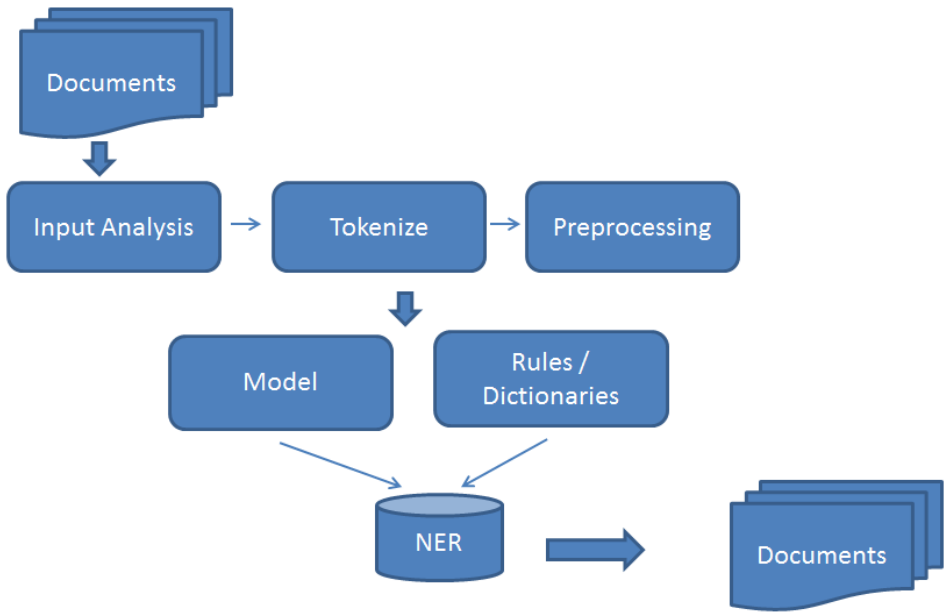

Figure 5. NER systems' architecture.

Figure 5 shows the general structure of the traditional NER systems. The system starts with documents as input (text in general); the input is analyzed and formatted to match the system's prerequisites and then converted to token-form. Preprocessing is applied to the formatted input adding specific system related features. The system then performs the recognition based on the trained model, rules and dictionaries; and outputs the predictions to documents similar in form to the input.

Due to the context of the project, the developed system had to be developed as an integrated system from scratch using Microsoft technologies stack for maintainability and integrability within the company's existing infrastructure of tools. Similarly, due to the need for a proprietary system, the majority of the modules had to be implemented from scratch. For the machine learning engine, C# was chosen as the main programming language with the integration of some low-level C libraries. The code was organized into classes referencing the different modules of the system based on the functionality provided by each of the modules. A C# machine learning framework was used for the statistical prediction implementation as well as an open-source implementation of the main CRF framework (CRF++) used by the majority of the systems in the literature. As a consequence to the nature of data and its volume preventing most of the datasets from being fully loaded into memory; streaming, splitting and buffering utilities were implemented to support the reading and the writing of the large inputs. To expose the functionality of the engine, a tabular user interface was designed to follow the functionality distribution of the system and enable access to the main functionality with ease.

Once implemented, the developed engine was hosted on a workstation with multiple CPUs of multiple cores and adequate memory to accommodate for the resource-heavy CRF training. The resource demanding aspect of the core engine was the reason for this setting. The functionality of the engine was locally exposed through an executable that gets installed on the end-user's machine and through a Web service to a planned-for Web application.

The developed NER system is composed of the preprocessing, CRF training, recognition, performance and postprocessing modules, as well as an initial tokenizer. Each of these modules has sub-modules and sub-functionalities that will be described in the following section.

3.2. Named Entity Recognizer Modules

3.2.1. Tokenizer

The first developed module of the system was an adapted tokenizer (lexical analyzer). Tokenization consists of converting any type of input into token form; a token can be a word, a number, a punctuation mark or an abbreviation. There are different approaches to handling this, many closely related to the target language, the specifications of the system and the input format desired or accepted by the other modules. In this context, tokenization means the splitting of a sentence into lexically and morphologically distinguishable tokens. In English, tokens are easily distinguishable since a blank space is considered as an almost definite word separator. Apart from a blank space, different systems have different approaches to tokenization where punctuation marks, numbering and normative designators are considered as word separators [Marrero et *al.*, 2013]. However, some systems choose to remove these markers and not consider them as tokens. This research opted to keep the delimiters and regard them as tokens because of the nature of the chosen paradigm, the nature of the target language and for uniformity between input and output. In some systems, tokenization is also used to classify analyzed tokens under predefined categories. Since the developed NER system includes a preprocessing module, classification was ultimately handled after the tokenization to keep the tokenizer language-independent. This held for languages that have a blank space as a rigid word separator; for other languages that do not have space separated words, the tokenizer includes an option to define specific word delimiters.

Another aspect of tokenization is the marking of the sentences, since in the context of text processing the sentences are regarded as the relevant sequences that from the context in which each token will be evaluated. Within our system, the sentences are the sequences that the CRF model will be trained on. End of sentence delimiters are crucial in the context of text analysis. This was the reason for which the implemented tokenizer paid close attention to marking the sentences. For simplicity, the system chose to mark the sentence by empty lines between each sequence of successive tokens.

The developed system's tokenizer takes multiple text formats as input, analyzes the data format and makes a tokenized output in the format of a text file (or a string list passed to other modules) that has one token per line and sentences separated by a new

empty line. This was achieved by specifically designed string splitters and by the use of regular expressions.

3.2.2. Preprocessing

The preprocessing module handles all processes related to data formatting. In addition, it handles adding relevant information to each token of the processed datasets; making the data ready for the different processes of the system. Preprocessing includes performing both language-independent and language-dependent lexical and morphological analysis, whitelist and lexicon analysis and matching, as well as adding relevant features and data format verifications.

The first step of preprocessing is reading the input in the form of sentences separated by a line break character. For the sake of this research, all input is in text file format and each sentence is in a separate line. For large datasets, the input is either streamed or split into manageable chunks that are loadable into the machine's memory. For streamed datasets, data is read and processed line by line until the end of the input. The two options were used interchangeably depending on the target task (formatting for training, formatting for testing, balancing sets and so on). The module then calls the tokenizer to convert the sentences into token form. The main goals of this module are:

1. Making sure the datasets are formatted into the standardized format that is accepted and unified for all other modules of the engine.
2. Adding the automated features and allowing the addition of language- and dataset-specific features with ease.

The standardized data format that the system follows is drawn from the conventional CoNLL 2003 data format, where each line within the dataset is composed of the respective token, its characterizing features separated by a specific delimiter (white space or a tab) and the label. Each sequence of tokens (a sentence) is then separated by an end of sentence delimiter. Figure 6 represents a sample sentence with the first column composed of tokens, the second of a token-characterizing feature and the third of a label. In this example, the feature is a lexical analysis and it has 3 characterizing values: C for capitalized tokens, P for punctuation marks and O for the other types of tokens.

```
Albert→C→B_PERSON
Einstein→C→B_PERSON
took→O→S
the→O→S
entrance→O→S
examinations→O→S
for→O→S
the→O→S
Swiss→C→B_ORGANIZATION
Federal→C→M_ORGANIZATION
Polytechnic→C→E_ORGANIZATION
in→O→S
Zürich→C→S_LOCATION
.→P→S
→|→
```

Figure 6. Sample data format.

The automated features added to every dataset include language-independent lexical analysis which is represented by analyzing the lexical form of each token and categorizing it into an object, a punctuation mark or a number. For languages with capitalization, a marker for capitalized tokens and the normalized form of the token are added as features. These are the first features added to each dataset and are crucial to the training of the CRF model for the processed dataset. Language-specific features are also added at this stage by the module's responsible processes. Language-specific processes perform the matching of each token to its corresponding obtained feature resulting from language-specific hand-made rules or from running the set through labelers, stemmers or any other external engines. An example would be, a stemmer used for the Finnish language to get the basic form of the token without the word ending.

Other NER related features are also added depending on the dataset processed. For example, one of the most widely used and most agreed upon feature for NER are the Part of Speech (POS) tags for each token. To obtain these, the system opted either for adding them manually by a linguist by matching the tokens to their corresponding tags from the corpus; or running the set through a POS tagger for the target language then adding the result as a feature to the set. The lexicon analysis also produces features that are added to the set for some types of data and tasks. After the matching and the evaluation of each token either as a standalone or as part of a composite entity; the lexicon features are added and can include noun markers, a "supposed to be capitalized" feature (for noisy data), the stem or the normalized form of words or the token

frequency within the set. Depending on the set to be processed, other features can also be added in aims of refining the language-specific or the task-specific characterization.

This module also handles formatting of the testing and validation sets by stripping the label from each row in the data-formatted corpus. The testing sets are datasets from the corpus that have the same data format but are not supposed to have a label part. Therefore, every row in the dataset is only composed of the token and its features as the goal of the system while processing testing sets is to add its own labels to the input. In addition, another functionality handled by this module is splitting corpora and balancing the sets. As will be seen in later sections covering the datasets, in supervised learning for NER the training set needs to be balanced in terms of the distribution of NEs across the set, more so than other sets. The corpus also needs to be split according to the conventional fashion in the field, where the entirety of the data is split into a training set having around half of the data, and testing and validation sets sharing the other half. The preprocessing module within our system handles these processes with predefined implemented functionality adjustable by different options.

3.2.3. CRF Training

This module is responsible for training CRF models based on the input training set obtained from the tokenizer and the preprocessing modules. The module takes as input the formatted training set composed of tokens, their corresponding features and their labels. L-BFGS [Byrd et *al.*, 1995] was used for this project to solve the feature-learning parameter covered in Subsection 2.2.4.

Figure 7 shows a sample sentence form the training set. The sentence is presented in token form with the resulting features from the tokenizer and preprocessing modules. In this example, and similar to the sample from Figure 6 the first feature is the lexical analysis with the same values (C for capitalization, P for punctuation, O for other objects); the second feature is POS tags (to be covered in later sections).

```
Albert C NNP B_PERSON
Einstein C NNP B_PERSON
took O VB S
the O NNP S
entrance O NN S
examinations O NN S
for O IN S
the O DT S
Swiss C NNP B_ORGANIZATION
Federal C NNP M_ORGANIZATION
Polytechnic C NNP E_ORGANIZATION
in O IN S
Zürich C NNP S_LOCATION
. P PUN S
```

Figure 7. Sample training data.

By reading the training data, the module builds the observation on the sequences represented by the sentences marked by the end of the sentence delimiter. Each row of the sequence is composed of the token residing in the first column, its corresponding features represented by all other columns of the row except the last one which is the label. Figure 7, represents a sample sentence from the training set. The first column has the tokens; the second, the automated lexical analysis; the third, the POS tags and the last one has the label. The module then goes though the CRF probability calculation, for each token X having a label Y for each sentence in the training data, serializes the binary features and exports the findings as explained in Subsection 2.2.4. The result is a CRF trained model. The implementation of this module was carried out using a combination of the Accord.net machine learning Framework [Roberto de Souza, 2010] for creating the distributions and the CRFShap implementation of CRF using .net C# [Fu, 2015]. CRFSharp uses a C implementation for L-BFGS to solve for the feature-learning parameter and is based on the reference implementation of CRF in C++ called CRF++ that is used by many NER systems in literature [Benajiba et *al.*, 2008; Silva et *al.*, 2006; Chiong and Wei, 2006].

The implementation uses parallelism and threading to take advantage of the multi-core characteristic of the workstation where the developed NER engine is hosted. However, most of the code is CPU-based and does not need a graphics card for processing. Consequently, the engine is usable in virtually any decent machine, though the variance in performance in terms of training capacity and training time is evident from computer to computer. The CRF model is trained on an N-gram representing the

distribution of each token in a sentence. In other words, every possible permutation of a sequence is considered to build the input observation which in turn is used to infer the output. This involves heavy calculations, which can have high demands for time and space depending on the size of the training data. Furthermore, the module handles the tweaking of the different parameters related to the CRF implementation. For example, to control the size of the trained model, a frequency shrinking parameter can be set to ignore all tokens having less than the set threshold value in frequency within the set. The threshold was between 0 and 100%; any values that were less than 1% in frequency were ignored.

3.2.4. Recognition

The recognition module takes as input the trained CRF model and the input to be labeled. The input can be in the tokenized form having the same structure as the aforementioned testing or validation sets, or it can simply be raw sentences. In the case of raw sentences, this module calls the tokenizer and the simple lexical analysis from the preprocessing module to construct rows with tokens and their corresponding automatic features. After the possible formatting, the input undergoes the recognition process where the probability of each token having a certain label is evaluated using the CRF model and depending on a tolerance threshold, the labels are added for each token. For each token within the input, the probability is computed and a confidence value is generated along with the probable label based on inferring from the trained model. The confidence value was based on the probability and the tolerance was set to 90% or more. Depending on whether the confidence value falls within the tolerance threshold, the token is recognized as an NE and marked with the corresponding label referencing the target class of the classification task and the trained model. For example, for a model trained on person, location and organization classes the corresponding label for each token will be either one of these target labels or a label stating that the token does not belong to any of the mentioned classes.

This module is responsible for producing the first output of the recognition process within the Hybrid NER paradigm. Raw CRF predictions are then either exported as such or go-on to undergo other processes provided by the postprocessing module.

3.2.5. Postprocessing

The postprocessing module handles operations performed on the raw CRF recognized data. Within this module, the rule-based NER functionality is implemented in the form of language-dependent grammar and context rules. The input is processed either in sentence or token form and rules are applied to it, resulting in refining the output when combined with the raw CRF results. The implemented rules at this stage are basic grammar and context rules where a label can be assigned based on the position of the word within the sentence. An example of the implemented rules for English is: if a token or a group of tokens are marked as an NE of the type Person or Location and the structure is preceded by one of the prepositions “in”, “on” or “at” the label of the NE is always Location. The rules are language-dependent, task-specific and optional.

The other main component of this module is the lexica analysis implementing the dictionary-based NER. Within this component, lexica of target NEs are compiled from various sources and are used to perform the matching to the input by analyzing the input and matching of each token or group of tokens against the existing NEs of the lexica. For our system, the lexica evolved from stage to stage and depending on the targeted NEs of the task; details of the lexica and lexica compiling are covered in later sections. When combined with the raw CRF results, a label selection mechanism between the results of the matching and the CRF outputs was needed. The system preferred to favor the machine learning approach for the recognition to support language-independent recognition. Consequently, the postprocessing module implements a selection mechanism to determine the final labels for the hybrid NER [Ahmadi and Moradi, 2015]:

- A token was not tagged as an NE by the CRF model => no change.
- A token was tagged as an NE by the CRF model, but was not tagged as such by the lexica matching => the CRF label is kept.
- A token is tagged as an NE by both CRF and lexica matching => the lexica label is kept.

To implement this functionality, a lexicon reader and tagger were developed allowing for efficient reading and loading of the NEs from lexica into optimized lists

that handle the matching better. This was also needed due to the amount of data that can be involved in the lexica. Given the fact that the lexica lists can be very large, up to millions of entries, the matching represents a classical string list matching algorithmic problem where the bigger the lists are, the more time it takes to search or match their corresponding elements. The traditional "naïve" method of using nested loops to compare each element within one list to each of the elements of the other list was not an option for our system due to the size of the input and the size of the lexica lists. Using this approach, given a sentence of **n** tokens and a lexica of **m** entries, the complexity of the matching would go as high as **O(nm)** which with large sizes of either **n**, or **m** would be absolutely impractical if possible at all. Therefore, we opted for a searching and matching method adapted to our case. The following is a concentrated summary of the used method:

- The lexica are tagged and aggregated into one list.
- The list is sorted and split into "buckets" based on the first letter of the NE (the NE can be a single word or a composite entity) C# hash sets were used instead of lists to make the search faster.
- The input is processed in sentence form and token permutations are created for each sentence to handle composite entities.
- For each permutation, we compare the first letter to the corresponding bucket.
- If the bucket size is under a certain threshold value (1000 items), we compare the input only elements to of that specific bucket.
- Otherwise we sort the bucket and split it into sub-buckets based on the *n* first letters. We then repeat the process of comparing the permutation's first *n* letters until the comparison bucket's size is less than the threshold value.

Using this method, we made sure that each input permutation is compared to a much smaller list of NEs. This greatly reduced the processing time, lowering the complexity to **O(n+k+p),** where **k** is the size of the comparison bucket/sub-bucket and **p** the size of the sentence permutations.

3.2.6. Performance

The performance module is responsible for computing the performance metrics covered in Subsection 2.2.8. It takes as input a dataset that has the same structure as the testing set with labels (gold data) and the output of the same set without labels after undergoing the recognition and label selection. It then compares the golden standard label (the original valid label from the gold data) against the label predicted by the system, resulting in the generation of the system's performance metric reports for the processed set. The module handles generating details of all the computed counts involved in determining how well the system is performing along with the calculation of the conventional evaluation measures and the corresponding detailed statistics.

4. Experiments and System Phases

During this research, a modified Scrum methodology was used where the progress was demonstrated in daily stand-ups along with the next steps. Due to the nature of the system the daily progress covered specific details of the implementations and the challenges encountered. The main progress was demonstrated in the two-week scrums. The project had multiple phases: firstly, it started by building the core of system and setting up an English experiment to evaluate the engine based on the machine learning approach alone. The second phase consisted of analyzing the performance of the system and introducing the postprocessing module's functionality by implementing the hybrid NER paradigm. The project then explored the performance of the system in unedited-noisy data by a mock participation in the Coling 2016 2nd shared task [Ritter et *al.*, 2016] followed by all the improvements to the system needed to carry out decent recognition in user-generated text. In parallel, scaling to other languages was carried out and datasets for languages other than English was laid out. Finally, planning and the initial implementation stages of the Web solution were carried out.

4.1. Experiments and Datasets Description

Machine learning relies heavily on data to train the recognition models. Since the system opted for the supervised learning approach for the machine learning component, the critical part of the project was to find data suitable for NER; data that are referred to as the corpus from which the main datasets are drawn depending on the exact purpose and task for which the system will be trained. Following the conventions within the field the data is split into three main sets: a training set used to train the recognition models, a validation set used to fine-tune the training parameters and a testing set used to measure the performance of the trained models. For the initial stage of the project the collected corpus was split into the training, validation and testing sets as detailed in Section 4.2. The training set was used to train the English CRF model on the three target classes: person, location and organization. The validation set was then used to fine-tune the training parameters for the trained model. The tuning parameters included [Fu, 2015]:

- The maximum and minimum number of model training iterations before ending the training (consolidating parameter learning convergence), the maximum was set to 1200 and the minimum to 3.
- The maximum number of words in a sentence; initially set to 150.
- The minimum token and feature frequency within the set, any feature or token occurring less than 10 times in the set was dropped.
- The minimum difference value of the obtained probability where if a value is less than 0.0001 between three consecutive training iterations, the value is accepted.
- The confidence value, which was set to anything more that 90%.

The testing set was used to evaluate the performance of the trained model as detailed in Section 5.1.

For the second phase of the project, the same training and testing sets from Phase I were used after rebalancing. There was no need for the validation set within this experiment as the model was tuned. Processing the testing set yielded the results detailed in Section 5.2 for the same target classes (person, location, organization) as in Phase I.

For the noisy data improvement stage, the only available sets were training and testing sets. Further description for the datasets modification and target classes in Section 4.4 and details of obtained results in Section 5.3.

The three experiments followed a similar workflow, where the data is formatted, the sets are formed, the models are trained, the recognition is performed and the performance is measured. Figure 8 represents the general overflow and the way the datasets were used in the experiments. After the datasets are formatted to the acceptable format by the system's modules (Subsection 3.2.2) the training set is kept as is and is used to train the CRF model. The testing and verification sets are stripped of their labels and two versions of these sets are kept, one with the gold standard labels (labels from the original corpus) and one without labels. The validation set, when used, is then processed using the recognition module of the system which results in a labeled set. The performance on the validation set is measured and depending on the results the model

parameters are tuned. The testing set undergoes the recognition step, the performance is measured by comparing the resulted labeled set to the gold standard and the experiment's results are obtained.

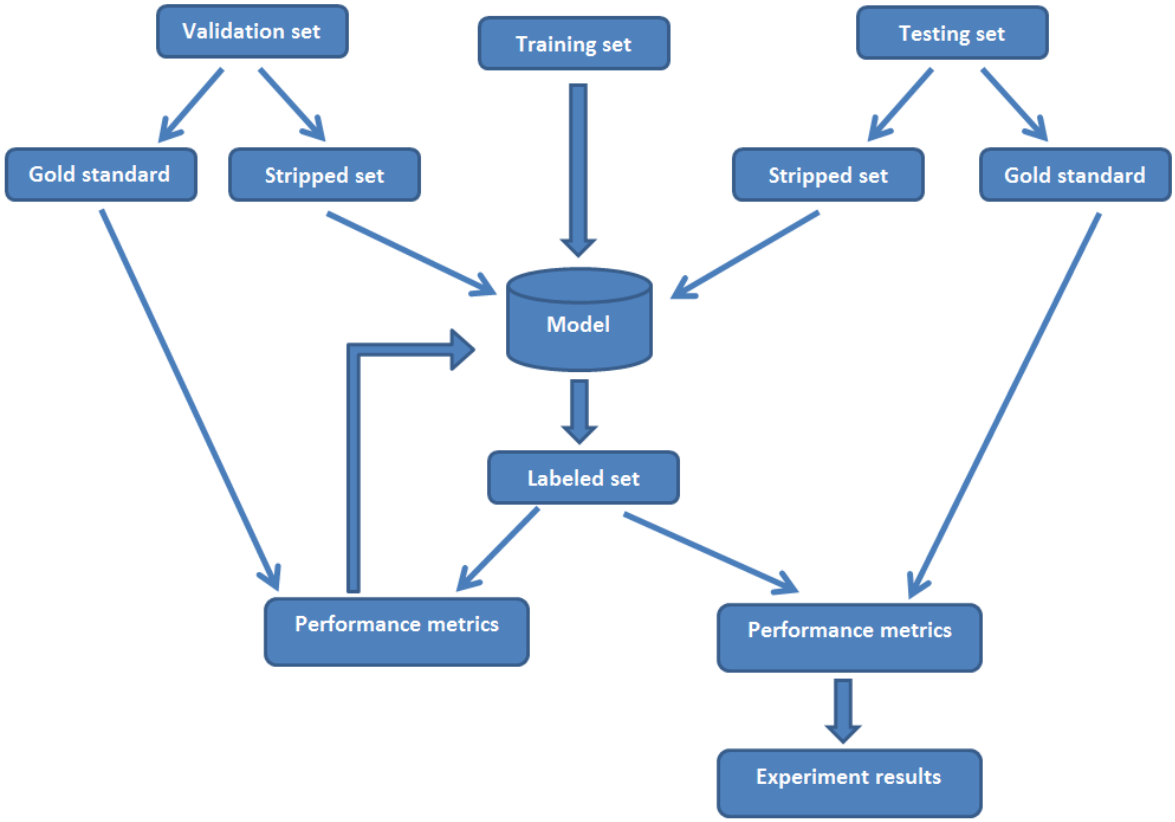

Figure 8. Experiment workflow.

4.2. Phase I: English Core

Since English was the target language to develop the core of the engine and for measuring the performance of the system, proper English corpus had to be gathered, organized, balanced and split into the main datasets used for most information extraction systems for the initial development tasks. The main corpus was put together in stages and from various sources. The abundance of English corpora played a pivotal role in getting a decent amount of data without major efforts. Data used for training machine learning engines are referred to as gold data. This type of corpora is manually edited and linguists are the main source of classification of the tokens. Gold data is the best data to train recognition models as it provides accurate information upon which the observations are made. Given a sentence, the linguist will mark the tokens with the

corresponding relevant label based on the context of the sentence and the role the relevant token holds within the sentence. The developed system needed NE-tagged datasets that have the following characteristics: sentences that have one or more of the target NEs marked in some form and a set that is large enough to train accurate models. The initial stage started by gathering freely available samples that satisfy the criteria, then formatting the samples to match the format that the training module accepted.

In the developed system, it was opted for a token-form where each line of the set was composed of a token followed by the features then the label. The collection started by freely available NER sets from news scripts tagged with person, location and organization NEs. The corpus was then expanded by adding sentences from the COCA corpus [Davies, 1990], which is a newspaper, popular magazines, fictional and academic text corpora available for commercial use. These additional samples went through the process of formatting, then through the Stanford NLP NER suite [Finkel et *al.*, 2005] after making our own Windows client to perform NER using this suite on the set and adding labels. The results were then formatted into a more readable form, one that is close to the format accepted by our training module. Manual verification and modification of the set was then carried out. The result was around 5.8 million tokens as detailed in Table 2. The system proceeded to balance the sets using the preprocessing module, since high-performance NER relies heavily on sets being balanced both in size and in distribution of NEs within each set. Following the conventions of the field, the sets were split into three main parts: a training dataset, a testing dataset and a verification dataset. The three sets were weighted according to the supervised learning approach conventions: half of the corpus went to training and the remaining half was split between the development and the testing set. Table 2 shows the distribution of data on the different sets, more than half of the sentences went to the training set, the rest went to the other two sets with the testing taking more sentences due to the core engine development evaluation.

Set	Sentences	Tokens	Entities	%
Training	84389	3504777	370383	51.86
Testing	53210	2002893	204548	29.64
Verification	34633	1250000	158819	18.50
Total	172232	6757670	733750	100

Table 2. Data distribution across datasets.

With the balancing of the sets, Table 3 shows the distribution of the data across the sets. The balancing at this stage involved having a distribution of the sentences across the three sets according to the conventional portion of sentences with each of the targeted classes (person location and organization) in each set of the sets.

For this phase, the main set that the in-set balancing (balancing the set itself) targeted was the training set. From Table 2, the 370383 entities were balanced for the three targeted classes using the preprocessing module's balancing function. A plus minus 5000 threshold was set (when possible) for the difference in number of entities of the different classes. Consequently, only 270383 were kept for the training (sum of the training row in Table 3).

Set	PERSON	LOCATION	ORGANIZATON	%
Training	89823	93808	86752	31
Testing	46037	82855	75656	30
Verification	21418	36460	70941	39
Total	157278	204548	158819	100

Table 3. Dataset Entity type balancing.

The NEs within the corpus were marked using labels detailed in Table 4, and for the composite entities, the system opted for marking each of the tokens within the composite entity with the entity boundary it represents: the first token was marked as beginning of the entity, the last token as the end of the entity and all the other tokens in the middle of the entity as middle of the entity. The target general classes for this phase were person, location and organization.

S	String
S_PERSON	Single PERSON
B_PERSON	Beginning PERSON
M_PERSON	Middle PERSON
E_PERSON	End PERSON
S_ORGANIZATION	Single ORGANIZATION
B_ORGANIZATION	Beginning ORGANIZATION
M_ORGANIZATION	Middle ORGANIZATION
E_ORGANIZATION	End ORGANIZATION
S_LOCATION	Single LOCATION
B_LOCATION	Beginning LOCATION
M_LOCATION	Middle LOCATION
E_LOCATION	End LOCATIOION

Table 4. Label set details.

For Phase I, the focus was mainly on the code behind the CRF training since the aim was to develop the CRF model training module. Only basic features were needed to evaluate how the model will perform with minimal information. Consequently, for this stage the feature set was only composed of the automatic lexical analysis feature added by the preprocessing module where capitalization and punctuation were marked. Capitalized tokens were marked with a binary value translated as a feature to a C, punctuation marks with a P and the other tokens with an O. Figure 7 illustrates an example of this.

At the end of this stage the system was ready to start the training of the English NER model, process the testing set (Figure 9), tune the model based on results and measure the performance of the trained model. In Figure 9, Each line has a token, a feature (Lexical analysis feature: C for capitalization, P for punctuation, O for other tokens)

```
Mark C
and O
I O
both O
use O
a O
variation O
of O
the O
5 O
row O
rule O
. P
```

Figure 9. Sample testing data.

4.3. Phase II: Analysis and Improvements

After the analysis of the initial results, the research focused on recognizing the problematic and challenging types and on the improvements needed to mitigate the observed limitations. The initial findings (Table 8 Section 5.1) showed problematic types where the entity boundaries proved challenging for the pure CRF model with simple lexical features. After extensive research of the probable causes, the findings yielded the need for adding POS tags as features for the set (Table 10 shows the improvement caused by adding POS tags), in addition to the simple lexical analysis of the tokens. Within the hybrid NER paradigm, the focus then shifted to the postprocessing module to improve the results by introducing a postprocessing step where each token within the processed set was compared and matched to "pure lists" compiled from Wikipedia for the target types. The lexica were composed pure lists of person, location and organization NEs that contained unambiguous entities without duplicates or noise. The lists did not include a large number of entities, but the refined NEs avoided ambiguous types and were certain to refer only to the correct type. The sets also had to be re-balanced, as after checking the data there were some underrepresented NE types. Specifically, the middle NE type within composite entities was poorly represented within the training set which manifested clearly in the initial results as shown in Table 8.

For this phase, the same main corpus was used with modifications based on the initial observation. The initial findings are detailed in the Table 9 in the research results section. Generally, the trained CRF model performed decently and the results matched similar metrics from the literature. However, there were some problematic types that required further dataset modification to improve the corresponding metrics. For the improvement phase, the same number of sentences was kept with the addition of extra sentences that included the previously underrepresented NEs and the removal of sentences that included NEs representing only the abundant types. Taking the example of the middle Person NE within the Person composite entities, the initial set had fewer sentences that included this specific entity; instead, there were more sentences that included a two-token composite entity with just a beginning and an end token.

Being a context-based feature that includes lexical, morphological and contextual analysis of the tokens, POS tags are widely used as a training feature in many information extraction applications [Benajiba et *al.*, 2008]. Since NER is one of them, and our system relying on the supervised learning approach, POS tags provided a rich feature that establishes the nature of the word, the context it holds within the sentence and some information about its morphology. Whether the token is a verb, noun, pronoun, conjunction, symbol, adjective, etc. has a crucial effect on refining the learning parameter and the feature function within the already covered structure of a CRF model [Benajiba et *al.*, 2008]. Consequently, for improving the metrics of the raw CRF prediction achieved in the first phase, the corpus was enriched by adding a POS tag feature to all the tokens. To achieve this, POS tags were salvaged from some of the previously collected free samples that included this feature for the sample data. Some of the COCA corpus had POS tags already added and for the rest of the data the Stanford POS tagger [Toutanova et *al.*, 2003] was used to add the corresponding POS tags. Figure 7 illustrates a sample of the corpus after adding the POS tags.

Consequently, after the English CRF model was retrained on the refined training data, the test set underwent the recognition process using the refined model. The next step in the improvements was to apply the processes of the hybrid NER approach by subjecting the resulted output to the postprocessing steps covered in Subsection 3.2.5. To do so, NE lists had to be compiled.

Given the targeted label set within this phase the research focused on compiling lists mainly from Wikipedia data dumps due to the availability of such data. However, with the nature of such data careful collection had to be carried out. Wikipedia data tend to have a considerable amount of noise and unfiltered data. In addition, the lists needed to be "pure" and only have distinct types to avoid ambiguity. Consequently, considerable efforts were made to compile the pure postprocessing lexica for the improvements phase. The task consisted of compiling lists of persons (first names and last names, celebrities and historical figures), locations (cities, countries and venues) organizations (universities, companies and NGOs) from DBpedia [DBpedia, 2007] using SPARQL [Sparql, 2008] Query language. The following sample query illustrated in Figure 10 was used to extract names of companies from the DBpedia data dumps

based of the ontology class name “company” within the dumps. That is, extract all strings marked with a link of the type company from the class set in the texts from the data dumps.

```
prefix dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?company
where {
  ?company a dbpedia-owl:Company
}
```

Figure 10. Sample SPARQL query.

The resulting lists were aggregated into the corresponding target label types and filters were applied to exclude all entries that had unrecognized or foreign characters, remove duplicates, URLs, redundancies and similar noise from the lists. The resulting lexica were then verified to refine the included NEs and make sure that the target labels are accurately represented. The next step was to use the postprocessing matching and lexica analysis methods to correct and refine the labels predicted by the CRF model.

The above improvements helped the system reach acceptable metrics. By the end of this phase, the system’s performance matched the state-of-the-art performance of the most recent literature on the Hybrid NER approach on similar corpora of edited data (“proper” sentences with capitalization, punctuations and context). The results are detailed and analyzed in the results section. Table 10 goes in details over the results and shows how the problematic type from Phase I were resolved and the overall performance of the system improved.

4.4. Coling Shared Task Mock Trial and Noisy Data Improvements

To further evaluate the developed system and compare its performance to research based and market existing systems alike, different datasets were used and different types of data were explored. Within this premise, came the mock participation in an NER shared task of Coling 2016. The Coling 2016, 26th International Conference on Computational Linguistics had a workshop on Noisy User-generated text and one task within the shared task was Named Entity Recognition in Twitter [Ritter et *al.*, 2015]. At this stage, the developed system was not suited for this type of data by any means. However, the chance was taken to evaluate the developed system and get the baseline

performances for this type of data. Hence, the research took part in the initial stages of the shared task up to the results submission as a good insight into this type of data and what might be involved in such tasks reported to be challenging for state-of-the-art NER systems.

Given these challenges covered in Subsection 2.2.9, tackling NER for noisy data must be adapted to suit the specificities of this task. Relying entirely on machine learning processes with conventional features to predict the labels is faced with the issues of contextualization and lack of formality. Inferring the context from such data is less accurate than in edited data. The same case applies to this research's developed system; in the first stages of the task, when faced recognition on noisy data, the trained CRF model on the data provided by the task's organizers yielded results that did not come even close to the performance of the system on edited data. The targeted types of this task included 10 fine-grained types that the system was not trained on. Consequently, the CRF model had to be re-trained using the given training data, which was not entirely suitable for CRF training as it was imbalanced, lacked any kind of features and had far fewer sentences (tweets in this case). The mock trial of the shared task stopped at this stage, as the main goal behind the participation was to evaluate the system and get initial insight into the challenging task of NER with noisy data. However, the research took this as a good opportunity to equip the system with new processes that would make it more suitable for future instances of this kind of data. A new experiment was set up to make improvements specifically tailored for dealing with noisy data. The next sections will describe the setting of the experiment and the results will be covered in the results section.

The data for the experiment were formatted using the system's preprocessing module and the lexical analysis feature was added. The data were collected by aggregating the training and testing sets from the Coling task [Ritter et *al.*, 2015], combined with some manually collected and annotated tweets with the original 10 fine-grained types. Figure 11 shows an example sentence from the Coling task sets which had no features, just the token and its label.

```
@emily B-person
rodriguez I-person
did O
n't O
bring O
my O
cookies O
today O
!! O
lil O
brat O
. O
https://gph.is/1Pwq4u3 O

```

Figure 11. Sample corpus of noisy data.

The number of tweets after the aggregation of the data from the Coling task [Ritter et *al.*, 2016] and the manually labeled tweets was around 2400 for the training set and 1500 for the testing set. Given that the data provided by the task were not to be redistributed, the sets were used strictly within the setting of this experiment and only to evaluate the system after the noisy data improvements. Two variants of the datasets were preprocessed and formatted: one variant was with 10 NE types (10 Types) and the other with NE types only determining the existence of an NE and its boundaries (No Types). The label set followed the same pattern as in the shared task, which also followed the CoNLL 2003 conventions. Tables 5 and 6 show the used label sets on which each of the two model variants were trained, and Figure 12 a sample sentence.

O	No type
B-other	Start of Other NE
I-other	Continuation/End of Other NE
B-geo_loc	Start of geo_loc NE
I-geo_loc	Continuation/End of geo_loc NE
B-product	Start of product NE
I-product	Continuation/End of product NE
B-facility	Start of facility NE
I-facility	Continuation/End of facility NE
B-company	Start of company NE
I-company	Continuation/End of company NE
B-person	Start of person NE
I-person	Continuation/End of person NE
I-sportsteam	Continuation/End of sportsteam NE
B-sportsteam	Start of sportsteam NE
I-musicartist	Continuation/End of musicartist NE
B-musicartist	Start of musicartist NE
B-movie	Start of movier NE
I-movie	Continuation/End of movie NE
B-tvshow	Start of tvshow NE
I-tvshow	Continuation/End of tvshow NE

Table 6. Label set for “10 Types” variant.

O	No type
B-other	Start of a NE
I-other	Continuation/End of a NE

Table 5. “No Types” variant.

Given the importance of the features in the training of the prediction and recognition CRF model, the feature set for this experiment was designed carefully to mitigate the limitations related to the nature of the training data. In addition to the initial lexical analysis feature, POS tags were included to provide valuable information on the word form, the context and the role it plays in the sequence. This, as demonstrated in the first phases of the practical part of the research, set a strong foundation for the recognition. However, to accommodate for the nature of noisy data, additional features had to be added. The additional features for the experiment on the noisy data are explained in Table 7.

Feature	Reasoning
Noun/Verb binary	This feature was derived from the POS tags and was added to provided additional information on the sentence structure.
Word frequency	This feature was used to compensate for the uniqueness of misspelled words, as the CRF model does not include them in the learning.
Normalized word	This feature was used to normalize tokens with @, # and other special symbols, as well as URLs and links, minimizing the noise for the CRF model.
Lexica matching feature	Arguably the most useful feature of all. It was used to add the results of the matching of the input to the lexica as a feature along with the type for the CRF model to learn from.
Supposed to be capitalized	Knowing how important capitalization was for English NER, this feature was used to mark tokens that are supposed to be capitalized in the input but were not capitalized.

Table 7. Feature set for noisy data.

These features were added to the datasets of the noisy data experiment, with the preprocessing module taking advantage of the implemented easy feature extension functionally. Further features were also incorporated for this round of improvement. The feature adding functionality was extended to use the previously described matching method (Subsection 3.2.5) to add the lexical matching feature. Accordingly, after the matching of the input to the lexica (more inclusive lists were added and are described in the next component of the experiment), if a token or group of tokens were found in one or more of the lists, the binary feature marked the token as such with the additional information of which types it matched. The preprocessing module was also modified to include functionality supporting the adding of the "supposed to be capitalized" feature. This feature marks tokens that are labeled in the corpus as NEs but are not capitalized. For example, if in the corpus we have the following sentence: "meet u @ Tampere airport", and "Tampere airport" was marked as an NE, the "airport" token will be marked as "supposed to be capitalized" (**SpCp**), as it will be in edited data. Figure 12 shows a sample sentence containing the added noisy data features.

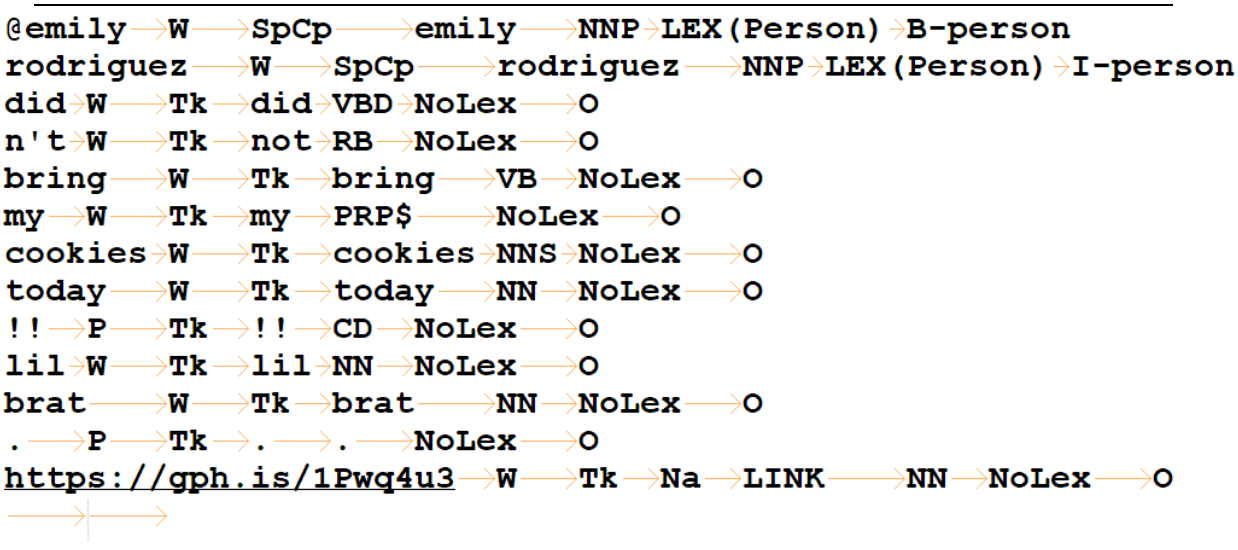

Figure 12. Sample training data for noisy data after processing.

Similar postprocessing techniques were applied to noisy data with some small modifications. The language-dependent grammar rules were still applied to the sentences after the initial matching. However, given the nature of the additional data, lexica had to be added to the postprocessing. Given that the corpus was labeled with 10 different types (Table 6), similar types had to be reflected in lexica. Consequently, additional lists of the corresponding types were collected and cleaned from noise using the existing cleaning functionality. The lexica included (reflected the target classes): artists, brands, companies, facilities, locations, movies, organizations, products, shows, songs, sports teams and a list for "other" NEs. Within this approach and keeping the expandability of the lists in mind, a configuration file was set up that held information about the target labels and the corresponding lexica that held the NEs of that type. Given the sheer size (the aggregated lists were up to 15 million entries) and the verity of the lists, fine-tuning these lists to be "pure" was not feasible. Instead, the lexica analysis method took into consideration only NEs that were unique for label correction as described in the lexical comparison and analysis method in Phase I, in what can be seen as a very basic form of the lexica matching and analysis.

After applying all these changes and improvements, the system was ready to train the CRF model using the sets adapted for noisy data, perform the recognition on the testing set, do the postprocessing and generate the final output. The results of the experiment are detailed in Section 4.2 in Tables 11 and 12.

4.5. Language Scaling

In parallel, with the noisy data improvements, scaling the system to other languages was carried out. Raw text corpora for French, Italian, Spanish, German, Finnish and Russian were extracted from Wikipedia data dumps salvaging resources from the WikiMedia project [Wikimedia, 2003]. The data dumps are composed of sentences that have links marked with specific patterns. After some exploration and cleaning of the dumps, where sentences were extracted using the implemented functionality within the preprocessing module, around 200000 sentences were extracted from the data dumps along with lexica for the usual person, location and organization NE types for all the mentioned languages. The next step was to prepare and generate NER-suitable corpora from the sentences. The characteristics for supervised learning corpora had to be respected and followed. The first two languages that the research started with for making NER corpora were Finnish and Russian. Given the size of the dumps, the downloaded files had to be streamed and split into manageable chunks, which was handled by the implemented streaming, buffering and splitting utilities of the system.

Working in collaboration with native linguists of the two languages, patterns within the data dumps were recognized and the corresponding scripts to salvage those patterns were implemented within the preprocessing module of the system. In addition, language specific features were extracted and added to the sets when applicable and when available in the respective language data dumps. For example, for Finnish the stem of the word without the word endings was present in the corresponding labeled NEs; therefore, the stem of the word was kept as a feature for the Finnish corpus.

The observed patterns were that within the cleaned sentences, the Wikipedia link categories marked the corresponding strings with the class name of the link from the defined ontology within Wikipedia. This served the purpose of the research very well. The relevant class names were defined by the native linguist and the corresponding sentences were extracted. For example, to extract sentences with the person NEs, sentences with patterns designating actors, athletes, singers, engineers, physicists, first names, forenames, last names, etc. were extracted. The extracted sentences were then balanced so that the sets would be composed of an equivalent number of sentences representing each target NE.

Given the fact that the core of the system is language independent, at the current stage of the research, the Finnish and Russian corpora are ready for starting the training of the CRF model and performing the recognition.

4.6. Service Oriented Architecture and Web Solution

To expose the functionality of the developed engine, a Web solution is intended to integrate the company's portfolio presented in the form of a machine intelligence portal providing various tools to support this trend. However, for NER, priority was given to the development of the actual core engine, given the nature of the task at hand and the technical settings of such systems. At the current stage of the research, a simplified UI is provided to the users of the engine. Given the fact that the system requires considerable resources to perform its functionality properly and efficiently, similar to most systems of the kind, the core engine is hosted on a powerful machine that can carry heavy processing loads in terms of memory and computing power. The functionality of the system is then exposed to different clients through a communication medium that implements certain communication protocols that can be understood at all ends. This research, given the technologies used to develop the core of the system, utilized a Web service using the Windows Communication Foundation (WCF), implementing a Service Oriented Architecture (SOA) that would support communication between the developed core and the Web tool, along with various other probable clients. The Web service is based on exposing the main functionality of the core engine and is responsible for establishing and managing the messages between the communicating ends. The Web tool will handle users and metadata management as well as reporting, but all actual functionality will be handled by the core engine of the system.

5. Research Results

After covering the system theoretical framework, literature, description of components, the system phases, experiments and the evaluation metrics used to measure the performance of the system for the different datasets, phases and the experiments; the next sections will go over the obtained results and provide brief summaries and interpretations of the results, since the analysis and explanation of the results were covered as the related component was explained in previous sections.

5.1. Phase I

The initial phase of the system was aimed at the development of the system's core CRF training. The training set covered in Section 4.2 was used to train the CRF model and the validation set was used to fine tune the training parameters. The processing of the testing sets (labeled and gold standard variants) and the comparison of the obtained labels and the gold standard yielded the following results for the initially targeted classes:

	Accuracy	Precision	Recall	F-measure
_PERSON	99.15%	75.91%	81.20%	78.46%
PERSON	99.57%	76.23%	83.05%	79.49%
B_PERSON	99.30%	80.25%	84.86%	82.49%
M_PERSON	99.96%	55.42%	65.49%	60.03%
E_PERSON	99.30%	80.25%	84.86%	82.49%
_LOCATION	99.02%	87.76%	90.40%	89.06%
LOCATION	98.84%	75.92%	84.50%	79.98%
B_LOCATION	99.77%	80.46%	85.84%	83.06%
M_LOCATION	99.95%	72.36%	76.11%	74.19%
E_LOCATION	99.77%	80.46%	85.84%	83.06%
_ORGANIZATION	98.92%	81.95%	85.34%	83.61%
ORGANIZATION	99.80%	76.29%	78.60%	77.43%
B_ORGANIZATION	99.77%	83.27%	86.77%	84.98%
M_ORGANIZATION	99.59%	81.41%	87.18%	84.19%
E_ORGANIZATION	99.77%	83.27%	86.77%	84.98%
Single Entities	99.40%	76.14%	82.05%	78.96%
Composite Entities	99.69%	77.46%	82.64%	79.94%
All Entities	99.03%	81.87%	85.65%	83.71%

Table 8. Detailed Phase I results.

Analyzing the results from Table 8 representing the raw CRF predictions on Phase I data (main initial English corpus with the initial datasets) showed problematic types, though most of the entities performed decently well with the F-measure ranging between 79% and 85% and an overall F-measure of 83%. For reference, processing similar sets with Stanford NER, which is CRF-based as well, yielded a similar F-Measure of 84% for the whole set [Finkel et *al.*, 2005]. This showed that the CRF training module was working properly, that it was learning from the training data and could predict labels based on the observations. However, the dips in performance for specific types such as the M_Person entity were undesirable. Issues discovered during this stage were mitigated during Phase II of the project.

5.2. Phase II

During this phase, the research's focus shifted towards identifying the issues discovered in the results detailed above, and on improving the metrics of the system. As covered before, the improvements included rebalancing the sets to recalibrate the underrepresented NEs, refining the used feature set and introducing postprocessing steps to implement the Hybrid NER processes and thus refine the raw CRF predictions. During this phase, two experiments were carried out: the first one was just rebalancing the data and postprocessing the prediction results, while the second further included adding POS tags. Thus, two experiments were designed to demonstrate the importance of features for CRF learning. Table 9 details the results obtained in the first experiment, where more sentences containing the underrepresented entities were added to the training set (Sentences with composite person class with more than two tokens); as well as postprocessing the results using the lexica analysis and label correction. The results of this experiment showed an overall improvement in F-measure (ranging between 84% and 88%) in all entities as well as the mitigation of the performance dips for the specific types (M_Person) from Phase I. Table 10 shows the results of the second experiment on the same balanced set with the addition of POS tags for training then postprocessing the prediction results using the lexica analysis. The results of this experiment show a considerable improvement in performance for all entities with F-measure scores ranging between 87% and 90%. These results are comparable to most results from literature on similar data.

	Accuracy	Precision	Recall	F-measure
_PERSON	99.78%	82.08%	89.24%	85.51%
PERSON	99.85%	83.05%	89.04%	85.94%
B_PERSON	99.90%	81.02%	88.67%	84.67%
M_PERSON	99.93%	81.07%	88.41%	84.58%
E_PERSON	99.94%	81.06%	88.16%	84.46%
_LOCATION	99.70%	88.81%	88.47%	88.64%
LOCATION	99.90%	87.47%	88.19%	87.83%
B_LOCATION	99.90%	87.83%	88.09%	87.96%
M_LOCATION	99.91%	87.77%	87.73%	87.75%
E_LOCATION	99.91%	88.02%	87.69%	87.86%
_ORGANIZATION	99.63%	88.97%	85.65%	87.28%
ORGANIZATION	99.90%	88.17%	85.19%	86.66%
B_ORGANIZATION	99.90%	88.00%	84.84%	86.40%
M_ORGANIZATION	99.89%	87.78%	84.67%	86.20%
E_ORGANIZATION	99.89%	87.81%	84.55%	86.15%
Single Entities	99.88%	86.23%	87.47%	86.81%
Composite	99.91%	85.59%	86.98%	86.22%
Entities ALL	99.70%	86.62%	87.79%	87.14%

Table 9. Detailed first experiment results.

	Accuracy	Precision	Recall	F-measure
_PERSON	99.79%	87.18%	90.40%	88.76%
PERSON	99.86%	86.17%	90.24%	88.16%
B_PERSON	99.91%	86.18%	89.85%	87.98%
M_PERSON	99.94%	86.20%	89.58%	87.86%
E_PERSON	99.94%	86.15%	89.32%	87.71%
_LOCATION	99.72%	93.19%	89.86%	91.50%
LOCATION	99.90%	91.93%	89.59%	90.74%
B_LOCATION	99.91%	92.25%	89.49%	90.85%
M_LOCATION	99.92%	92.18%	89.13%	90.63%
E_LOCATION	99.92%	92.40%	89.09%	90.71%
_ORGANIZATION	99.64%	93.30%	86.98%	90.03%
ORGANIZATION	99.90%	92.54%	86.55%	89.45%
B_ORGANIZATION	99.90%	92.36%	86.19%	89.17%
M_ORGANIZATION	99.89%	92.12%	86.00%	88.95%
E_ORGANIZATION	99.89%	92.14%	85.87%	88.89%
Single Entities	99.89%	90.21%	88.79%	89.45%
Composite	99.91%	90.22%	88.28%	89.19%
Entities ALL	99.72%	91.23%	89.08%	90.10%

Table 10. Detailed second experiment results.

By the end of this phase, the system reached satisfying performance metrics comparable to state-of-the-art research on Hybrid NER for similar datasets. Keeping in mind that the metrics depend greatly on the target task and quality of the data, precise

comparisons of such system can only be made with strict limitations on the available resources and the techniques employed. The developed system achieved the desired performance for the target language without serious problematic types or dips in performance for particular types.

5.3. Noisy Data

As explained before, this experiment aimed particularly at targeting noisy user-generated data. After getting an insight into the nature of the data and the label types utilized in such tasks, the research used the findings to equip the developed system with the basic necessary processes to handle noisy data. The experiment aimed exclusively at evaluating the system and the developed model will be retrained on better, larger proprietary datasets. The initial intent was merely to get a general idea of how well the improvements handled data of this nature. The experiment included training two models of two variants of the training set, one with 10 types referred to as "10 Types" model and another variant with "No Types" (just the existence of an NE and its boundaries) referred to as the "No Type" model. Following are the results of processing the two variants of the test set by the developed system (referred to as SCS_NER, or "Services for cognitive systems NER", which is the name of the system within the company's portfolio) and the provided baseline system results from the Coling 2016 shared task. The baseline system in this task was based on crfsuite [Okazaki, 2007] and used lexica lists for feature generation. The CoNLL evaluation script was used to generate the metrics. Table 11 details the results obtained in the "No Type" variant and Table 12 the "10 Types" one. Both tables show that our system after the noisy data improvements performed better in terms of precision and recall than the baseline system on the same dataset. There were types that our system handled better in terms of F-measure, a clear example is the person type; types where the baseline system performed better, such as sportsteam and products. There were also other types where both systems performed equally poor; namely, the tvshow type (with 0% in precision and recall) which was because the training and testing sets only had a small number of tokens of this type; consequently, neither system could recognize this type.

	Accuracy	**Precision**	**Recall**	**F-measure**
Baseline	95.01%	54.21%	49.62%	51.82%
SCS_NER	95.49%	68.45%	50.42%	58.06%

Table 11."No Types" model performance results.

	Precision	Recall	F-Measure
Baseline (Accuracy: 93.68%)			
company	64.44%	33.69%	44.24%
facility	34.63%	26.43%	29.97%
geo-loc	59.75%	58.00%	58.86%
movie	18.18%	9.76%	12.70%
musicartist	32.14%	13.92%	19.42%
other	38.89%	20.87%	24.23%
person	36.27%	35.51%	35.88%
product	12.68%	7.41%	9.35%
sportsteam	21.14%	9.52%	13.12%
tvshow	0.00%	0.00%	0.00%
Total	**45.30%**	**33.60%**	**38.58%**
SCS_NER (Accuracy: 92.67%)			
company	78.29%	29.35%	43.56%
facility	63.11%	57.14%	52.07%
geo-loc	77.84%	47.47%	60.33%
movie	16.67%	6.67%	9.52%
musicartist	50.00%	12.50%	17.20%
other	54.34%	25.51%	31.28%
person	59.65%	55.59%	57.54%
product	31.25%	4.63%	8.06%
sportsteam	50.00%	3.57%	8.79%
tvshow	0.00%	0.00%	0.00%
Total	**66.02%**	**32.25%**	**43.33%**

Table 12. “10 Types” model performance results.

The resulting metrics of the two systems showed that the developed system, after noisy data related changes and improvements, performed better than the provided baseline on the same dataset. The F-measure comparison showed that the system had an overall better performance than the baseline. However, for recall, the system performed slightly worse than the baseline, while the precision of the system was better. This shows that, for the same dataset, the developed system could recognize more NEs that were actual NEs than the baseline, but was getting some of the label types wrong. Further analysis showed that, with a refined postprocessing to correct the labels of the recognized NEs, the recall can reach optimal levels. This can be achieved by performing the lexica matching and analysis.

In addition, the results for noisy data processing showed how this type of data proves challenging to NER. Given the nature of such data, recognition is not as performant as in edited data due to the difficulties and challenges related to user-generated data discussed before in Section 4.4. This is still an active research topic within the field and improvements to techniques applied to noisy data is a topic of interest to many current natural language processing and cognitive science conferences and shared tasks. Consequently, improvements to the applied techniques are still to be implemented in the near future.

6. Conclusions

6.1. Summary and General Reflections

The presented research tackled an application within one of the most promising fields of the present information technology world. Machine learning is becoming a source of intrigue and interest for every company around the globe; all the big players of today's IT market are investing heavily in research to exploit the advantages that the concept offers. Every company, research center or university that aspires to take part in this trend is investing in and developing systems that rely on this concept. Knowledge discovery and data structuring is becoming a necessity within today's world, where knowledge is power and information is the new precious metal. The research explored an application within the field that is considered to be the base upon which bigger and more complex systems of the field rely on. Within this work, a named entity recognizer was conceptualized, designed, implemented, improved, and evaluated. The developed system implemented the best performance-yielding techniques within the field and the study conducted multiple experiments to evaluate the performance of the system. The drive of this research was the need for a proprietary system that is user-friendly, performant and language-independent. Within this work, state-of-the-art literature was reviewed and presented. The theoretical focus was on the techniques that claimed to yield the best performance metrics for NER in different settings. The research then moved to the implementation of the selected methods and techniques using the most suitable resources for the scope of the project.

Data play a major role in the machine learning field. Consequently, a good portion of the research focused on collecting, curating, formatting and organizing suitable data for NER. Corpora and lexica were used to train, test and improve the performance of the core of the system. Statistical prediction was used to equip the system with the ability to recognize and classify previously unseen input. To achieve this, the system made use of the best NER-suited characteristics of Conditional Random Field models. Though it is the core of the system, the research did not stop at the point where the system would rely entirely on statistical prediction alone, but went on to explore, implement and integrate different techniques and methods falling within the Hybrid

NER paradigm to achieve the best attainable performance of such systems and match the state-of-the-art literature on the subject.

Dealing with different Big Data related aspects was also among the focal points of this research. Having datasets that cannot be loaded into memory but still have to be processed, matching two huge lists of strings, cleaning noise from lists of millions of entries, exploring huge data dumps and extracting patterns and relevant information from them were among the many intriguing aspects of this research. With this kind of data, naïve methods are slow at best and completely unusable in certain cases. Tailored methods were demanded to mitigate the limitations of traditional data processing, methods that were sometimes well-known techniques in the field and sometimes the research's own interpretation and handling of the task.

To evaluate the developed system, the research conducted multiple experiments using different datasets, feature sets and postprocessing techniques. The performance of the system was measured for each of the experiments and areas for improvement were identified and worked on. The analysis of the research observed the decent performance of the system and the achievement of its set goals of good performance and a language-independent core. The main modules of the system remained language-invariant where reading, formatting and processing data supported various options when such independency was not possible. The core of the system, being based on CRF models, is also language-independent and the developed training and recognition modules can learn and classify input from different languages. Likewise, data for different languages were collected and techniques to curate it were developed making use of freely available data that can be used for future experiments and for scaling the system to different languages.

Different types of data were explored within this research, including edited data, which involved evaluating the system's performance on such data and improving it, as well as noisy data with their specific techniques and characteristics requiring adapted methods to be implemented. The results obtained showed the capability of the developed system to deal with different types of data conventionally used within the field with adaptability to newly encountered types. Throughout this research user-friendliness was kept in mind, and multiple means of exposing the developed engine

functionality, including a Web solution were either fully implemented or planned and founded.

At this stage, the research achieved the goals it was set to achieve to various levels of completeness. However, given the nature of project and the active evolution of the research field, some of the modules of the system merely scratched the surface of the full potential for some of the methods. Every few months, new findings are added to the field and new techniques and methods are explored. Consequently, the research can still be categorized as on-going and the work in the project will continue as it is important to the company that supported it.

6.2. Research Limitations

The research focused on studying only the best performing techniques to achieve NER within the field. This might be considered a theoretical focus that might have limited the research. However, due to the nature of the field and the abundance of literature, the only way to keep the scope of the project within reasonable limits was to select the apparently best techniques from the literature, focus on them and design experiments to evaluate the proof-of-concept modules of the system implementing them. Some of the techniques used only approach the basic form of the task, as there are more advanced methods and more complex combinations that can be used.

The project was largely limited by the availability of data. As referred to within the field, gold data are very hard to come by and even when available, they might not have complete information that the target task and the system are relying on. The reliance of supervised learning on large amounts of annotated data that are hard to find, domain-specific and reliant on human involvement, is restricting systems implementing this paradigm. However, it is the best resulting approach this far that makes the actual recognition mechanism language-independent in terms of implementation and lets the data alone determine what is to be learnt by the trained model.

The core of the system was evaluated using only English datasets at this point of the research. English is heavily studied in the NER field; literature, resources and data are abundant for this language. Consequently, the only viable evaluation of the developed system is for this language, as other languages have specific characteristics that might

make the training less efficient than in English. However, for comparison and evaluation of how well the developed methods measure against state-of-the-art in the field, English is the best candidate for any proof-of-concept.

Within this research field, exact comparisons of two systems cannot be held unless the same datasets are used. Since in most cases the datasets are specifically designed for a particular system, they are normally not made available, especially in the case of proprietary systems. Evaluation within the field of machine learning stays relative, as even evaluation methods differ from system to system; some systems might use partial credit, others might use type-based matching instead of spatial matching. Consequently, the evaluation of these systems is dependent on the actual choices of the research. Unless the same datasets are used and limitations on what methods to use are introduced, comparing two systems will not be an exact science.

The CRF training model is resource-demanding and time-consuming and cannot be handled with ease by a normal user-oriented machine. To train a CRF model on a training set that has millions of tokens and hundreds of thousands of sentences is proven to be computationally costly and can take hours, even days to train due to the sheer amount of calculations needed. In addition, the postprocessing also requires relatively long periods of time to perform as the matching lexica grow in size and include more types.

6.3. Future Work

The presented research is still on-going and work on the project is still planned and carried out. Improvements, scaling to other languages and further exposing of the provided functionality are the potential directions of future work in this project. The research field is very active nowadays, as every few months new methods are developed and the promise of improvements in challenging languages and types of data is prominent. Consequently, the project is still open to the integration of other methods for all the developed modules.

Work on the areas covered in the research limitations is the most pressing direction. Experimenting on techniques that might not perform as well for specific datasets but have characteristics that make them better candidates for different datasets will be a

good addition to the project. Exploring approaches that do not rely as heavily on data for training can be an excellent opportunity for the system to expand. A good example of this would be to explore the semi-supervised learning approach, where the system starts by a small amount of annotated data combined with more available, easier to collect unannotated data, makes use of the observations from the annotated data to annotate the unannotated data and to refine the trained model. This can be a very good candidate for refining the training module of the system for languages with limited resources.

Incremental learning would be also a potential further development direction. For complex training sets using incremental learning, where the model learns only from portions of the sequences at a time, then keeps retraining itself until the whole observation is completed, the process is claimed to save considerable training time. In addition, optimizing the matching methods to have even better performance as the matching lists grow will also be investigated.

The most immediate direction of future work in this research is experimenting with the system in different languages and evaluating the performance on languages known to be challenging. Collection and formatting of some languages is already carried out within the presented work, and the next step would be to train the models for these languages and reflect on the achieved performance. In addition, collection and formatting of other corpora of languages that do not have traditional characteristics, such as a white space delimiter as a token separator, or do not have capitalization would create new challenges and additional research opportunities for this project.

References

[Ahmadi and Moradi, 2015] Farid Ahmadi and Hamed Moradi, A hybrid method for Persian named entity recognition. In: *Proceedings of the 7th Conference on Information and Knowledge Technology (IKT)* (2015), 1-7.

[Aly, 2005] Aly, Mohamed. (2005). Survey on multiclass classification methods. California Institute of Technology. *Available at: https://www.cs.utah.edu/~piyush/teaching/aly05multiclass.pdf*

[Andrushchenko, 2018] Mykola Andrushchenko, discussion and review, 2018.

[Atdağ and Labatut, 2013] Samet Atdağ and Vincent Labatut, A Comparison of named entity recognition tools applied to biographical texts. In: *Proceedings of the 2nd International Conference on Systems and Computer Science* (2013), 228-233.

[Benajiba et *al.*, 2008] Yassine Benajiba, Mona T. Diab and Paolo Rosso, Arabic named entity recognition using optimized feature sets. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2008), 284-293.

[Bikel et *al.*, 1997] Daniel M. Bikel, Scott Miller, Richard Schwartz and Ralph Weischedel Weischedel, Nymble: a high-performance learning name-finder. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997), 194-201.

[Borthwick et *al.*, 2002] Borthwick Andrew, John Sterling, Eugene Agichtein and Ralph Grishman, NYU: Description of the MENE named entity system as used in MUC-7. In: *Proceedings of the Seventh Message Understanding Conference* (2002).

[Brownlee, 2016] Jason Brownlee, What is a Confusion Matrix in Machine Learning (2016). *AlgorithmsFromScratch*. [Online]. *Available at:* https://machinelearningmastery.com/confusion-matrix-machine-learning/

[Brychcin et *al.*, 2015] Michal Konkol, Tomaš Brychcin and Miloslav Konopik, Latent semantics in named entity recognition. *Expert Systems with Applications* **42** (7), 2015, 3470–3479.

[Byrd et *al.*, 1995] Richard H. Byrd, Peihuang Lu, Jorge Nocedal and Ciyou Zhu, A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific and Statistical Computing* **16** (5), 1995, 1190-1208.

[Chang et *al.* 2011] Fang Luo, Han Xiao and Weili Chang, Product named entity recognition using conditional random fields. In: *Proceedings of the Fourth International Conference on Business Intelligence and Financial Engineering* (2011), 86-89.

[Chiong and Wei, 2006] Raymond Chiong and Wang Wei, Named entity recognition using hybrid machine learning approach. In: *Proceedings of the 5th IEEE International Conference on Cognitive Informatics* (2006), 578-583.

[Davies, 1990] Mark Davies, The Corpus of Contemporary American English (COCA) (1990). [Online]. *Available at: https://corpus.byu.edu/coca/*

[DBpedia, 2007] DBpedia database project (2007). [Online]. *Available at:* http://wiki.dbpedia.org/

[Finkel et *al.*, 2005] Jenny Rose Finkel, Trond Grenager, and Christopher Manning, Incorporating non-local information into information extraction systems by Gibbs sampling. In: *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics* (2005), 363-370.

[Fu, 2015] Zhongkai Fu, CRFSharp Conditional Random Fields (CRF) implemented by .NET(C#) (2015). *Available at:* https://github.com/zhongkaifu/CRFSharp

[Gagné, 2013] Christian Gagné, Evolutionary computation for supervised learning. In*: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation* (2013), 827-844.

[Gao et *al.*, 2017] Yu Yuan, Jie Gao and Yue Zhang, Supervised learning for robust term extraction. In: *Proceedings of the International Conference on Asian Language Processing (IALP)* (2017), 302-305.

[Gassert, 2017] Alden Gassert, Vocabulary flashcards: Graph theory. Department of Mathematics and Computer Science Hobart and William Smith Colleges (2017). *Available at:* http://math.hws.edu/gassert/Flashcards/Flashcards-GraphTheory.pdf

[Grishman and Sundheim, 1996] Ralph Grishman and Beth Sundheim, Message understanding conference-6: A brief history. In: *Proceedings of the 16th Conference on Computational Linguistics* (1996), 466-471.

[Kanya and Ravi, 2013] Kanya Varathan and Vignesh T. Ravi, Machine learning based biomedical named entity recognition. In: *Proceedings of IET Chennai Fourth*

International Conference on Sustainable Energy and Intelligent Systems (2013), 380-384.

[Kedad et *al.*, 2007] Zoubida Kedad, Nadira Lammari, Elisabeth Métais, Farid Meziane and Yacine Rezgui (Eds.), Natural language processing and information systems. In: *Proceedings of 12th International Conference on Applications of Natural Language to Information Systems (NLDB)* (2007).

[Kripke, 1982] Saul Kripke, *Naming and Necessity*. Harvard University Press, 1982.

[Kuperus et *al.*, 2013] Jasper Kuperus, Cor Veenman and Maurice van Keulen, Increasing NER recall with minimal precision loss. In: *Proceedings of the European Intelligence and Security Informatics Conference* (2013), 106-111.

[Lafferty et *al.*, 2001] John Lafferty, Andrew McCallum and Fernando Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), 282-289.

[LaPorte, 2016] Joseph LaPorte, Rigid Designators. Stanford Encyclopedia of Philosophy, Stanford University, 11 Feb. 2016, plato.stanford.edu/entries/rigid-designators/.

[Luo et *al., 2012*] Fang Luo, Pei Fang, Qizhi Qiu and Han Xiao, Features induction for product named entity recognition with CRFs. In: *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (2012), 491-496.

[Marinho de Oliveira et *al.*, 2013] Diego Marinho de Oliveira, Alberto H. F. Laender, Adriano Veloso and Altigran Soares da Silva, FS-NER: A lightweight filter-stream approach to named entity recognition on twitter data. WWW 2013 Companion. In: *Proceedings of the 22nd International Conference on World Wide Web* (2013), 597-604.

[Marrero et *al.*, 2013] Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato and Juan Miguel Gómez-Berbís, Named entity recognition: fallacies, challenges and opportunities. *Computer Standards & Interfaces* **35** (5), 2013, 482-489.

[Masayuki and Matsumoto, 2003] Masayuki Asahara and Yuji Matsumoto, Japanese named entity extraction with redundant morphological analysis. In: *Proceedings*

of the Human Language Technology conference - North American chapter of the Association for Computational Linguistics (2003), 8-15.

[Meselhi et *al.*, 2014] Mohamed A. Meselhi, Hitham M. Abo Bakr, Ibrahim Ziedan and Khaled Shaalan, Hybrid named entity recognition - application to Arabic language. In: *Proceedings of the 9th International Conference on Computer Engineering & Systems (ICCES)* (2014), 80-85.

[Nadeau and Satoshi, 2007] David Nadeau and Sekine Satoshi, A survey of named entity recognition and classification. *Lingvisticae Investigationes* **30** (1), 2007, 3-26.

[Nadeau, 2007] David Nadeau, Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision. Faculty of Graduate and Postdoctoral Studies in partial fulfilment of the requirements for the PhD degree in Computer Science Canada, 2007.

[Neumann and Xu, 2004] Günter Neumann and Feiyu Xu, Machine learning for named entity recognition. LT-lab, DFKI German Research Centre for Artificial Intelligence. (2004).

[Nongmeikapam et *al.*, 2011] Kishorjit Nongmeikapam, Tontang Shangkhunem, Ngariyanbam Mayekleima Chanu, Laisuhram Newton Singh, Bishworjit Salam and Sivaji Bandyopadhyay, CRF based name entity recognition (NER) in Manipuri: A highly agglutinative Indian language. In: *Proceedings of the 2nd National Conference on Emerging Trends and Applications in Computer Science*, (2011), 1-6.

[Okazaki, 2007] Naoaki Okazaki, CRFsuite: a fast implementation of Conditional Random Fields (CRFs) (2007). [Online]. *Available at*: http://www.chokkan.org/software/crfsuite/

[Poibeau, 2003] Thierry Poibeau, The multilingual named entity recognition framework. In: *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics* (2003), 155–158.

[Poibeau, 2006] Thierry Poibeau, Dealing with metonymic readings of named entities. In: *Proceedings of the 28th Annual Conference of the Cognitive Science Society* (2006).

[Prasad and Fousiya, 2015] Gowri Prasad and K. K. Fousiya, Named entity recognition approaches: a study applied to English and Hindi language. In: *Proceedings of the International Conference on Circuit, Power and Computing Technologies [ICCPCT]* (2015), 1-4.

[Ram et *al.*, 2010] Vijay Sundar R. Ram, A. Akilandeswari and Sobha Lalitha Devi, Linguistic features for named entity recognition using CRFs. In: *Proceedings of the International Conference on Asian Language Processing* (2010), 158-161.

[Ratinov and Roth, 2009] Lev Ratinov and Dan Roth, Design challenges and misconceptions in named entity recognition. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (2009), 147-155.

[Ritter et *al.*, 2011] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni, Named entity recognition in tweets: an experimental study. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)* (2011), 1524-1534.

[Ritter et *al.*, 2016] Alan Ritter, Bo Han, Leon Derczynski, Wei Xu and Tim Baldwin (2016) The 2nd Workshop on Noisy User-generated Text (W-NUT): Named entity recognition in twitter. [Online]. *Available at:* *https://noisy-text.github.io/2016/*
https://github.com/aritter/twitter_nlp/tree/master/data/annotated/wnut16

[Roberto de Souza, 2010] César Roberto de Souza, Accord.Net Framework (2010). [Online]. *Available at: http://accord-framework.net/*

[Rouse, 2016] Margaret Rouse, What is Machine Learning? Definition from WhatIs.com, *WhatIs.co*. 15-Feb-2016. [Online]. *Available at: http://whatis.techtarget.com/definition/machine-learning.*

[Salama et *al.*, 2015] Khalid M. Salama, Ashraf M. Abdelbar and Fernando Otero, Investigating Evaluation Measures in Ant Colony Algorithms for Learning Decision Tree Classifiers. In: *Proceedings of IEEE Symposium Series on Computational Intelligence* (2015), 1146-1153.

[Satoshi, 1998] Sekine Satoshi, Nyu: Description of the Japanese NE system used for Met-2. In: *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[Silva et *al.*, 2006] Eduardo F.A. Silva, Flavia A. Barros and Ricardo B.C. Prudencio, A hybrid machine learning approach for information extraction. In: P*roceedings of Sixth International Conference on Hybrid Intelligent Systems (HIS'06)* (2006), 1-18.

[Sparql, 2008] SPARQL Protocol and RDF Query Language (2008). [Online]. *Available at:https://dbpedia.org/sparql*

[Tjong and De Meulder, 2003] Erik F. Tjong Kim Sang and Fien De Meulder, Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL* (2003), 142-147.

[Toutanova et *al.*, 2003] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer, Feature-rich part-of-speech tagging with a cyclic dependency network. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (*2003), 173-180.

[Wallach, 2004] Hanna M. Wallach, Conditional Random Fields: An Introduction. Technical Report MS-CIS-04-21. Department of Computer and Information Science, University of Pennsylvania, 2004. Also available *as https://repository.upenn.edu/cis_reports/22/*

[Wikimedia, 2003] Wikimedia Foundation, Wikimedia Downloads (2003). [Online]. *Available https://dumps.wikimedia.org/*

[Zuhori et *al.*, 2017] Syed Tauhid Zuhori, Asif Zaman and Firoz Mahmud, Ontological knowledge extraction from natural language text. In: *Proceedings of the 20th International Conference of Computer and Information Technology (ICCIT)* (2017), 1-6.